UNIVERSITE LIBRE DE BRUXELLES
Faculte Des Sciences
Departement D'Informatique

Toward a Brain-like Memory with Recurrent Neural Networks

Thèse présentée par

Utku Salihoglu

En vue de l'obtention du grade de

Docteur en Sciences

Novembre 2009

Acknowledgments

Je voudrais commencer par remercier sincèrement mon promoteur, Hugues Bersini, qui m'a permis d'entreprendre cette thèse et m'a également épaulé tout au long de cette longue aventure. Il s'est avéré, au cours des années, un repère indispensable tant pour ma recherche que pour mes autres activités à l'université. Un très grand merci à Colin Molter pour avoir partagé, durant ces six dernières années, cette recherche avec moi. Il a été un guide: ses idées, commentaires et remarques m'ont été d'une grande aide.

Je voudrais également prendre le temps ici de remercier Raymond Devillers, partie intégrante de mon cursus à l'université de ma première candidature jusqu'à ce jour, qui m'a poussé au cours des années à faire toujours mieux. Un tout grand merci tout particulièrement pour tous ses commentaires durant la rédaction de cette thèse ; du fond jusqu'à la forme en passant par l'orthographe, il n'a rien laissé au hasard. Je suis particulièrement reconnaissant à chacun des membres du jury de l'honneur qu'ils m'ont fait en acceptant d'évaluer mon travail. Ainsi mes remerciements vont à Gianluca Bontempi, Pierre Gaspard, Axel Cleeremans et Robert Kozma. Je voudrais également remercier les membres de mon comité d'accompagnement, Pasquale Nardone et Marceline Kaufman, pour m'avoir suivi durant les années qui ont été requises pour voir l'aboutissement de ce travail.

Je ne pourrais continuer ces remerciements sans inclure mes collègues et amis de l'IRIDIA. Tous, à leur façon, ont contribué à créer un environnement de recherche unique et extrêmement agréable. Un tout grand merci à Christophe Philemotte pour son soutien, ses remarques et son aide durant ces années, depuis le premier jour où j'ai posé le pied à l'IRIDIA. Un grand merci également à Amin Mantrach qui partage mon parcours depuis nos études en informatique et qui a toujours répondu présent. Je voudrais également remercier Bruno Marchal pour tous ses commentaires mais également tous ces sujets de discussion passionnants qui animent nos conversations depuis pas mal d'années maintenant. Comment remercier l'IRIDIA sans remercier le cœur qui fait battre cette machinerie complexe et qui permet à tout un chacun de trouver sa place, Muriel Decreton–Longheval.

Un énorme merci à Martin Van Aken, Gilles Vanwalleghem, Cédric Meiresonne et Jonathan Vanderick qui, en plus de leur soutien, m'ont également fourni moultes corrections et commentaires qui m'ont été d'une valeur inestimable. Je tiens finalement à remercier Christelle Thienpont, Jeanne Stemgée, ma mère et tous ceux que je n'ai pu nommer ici mais qui ont été une partie non négligeable de ma vie durant ces dernières années.

Utku Salihoglu – Bruxelles, Novembre 2009

Contents

Acknowledgments iii

List of Figures vii

List of Abbreviations
and Symbols xi

List of Abbreviations and Symbols xiv

Chapter 1. Introduction 1

Chapter 2. Background 9
1. Neurophysiology 9
2. The Physiological Basis of Behavior 15
3. Memory 18
4. Hippocampus 26
5. Connectionist Models 33
6. Dynamical Systems 41
7. Conclusion 62

Chapter 3. Model 63
1. Methodology and State of the art 63
2. Unit Definition 67
3. Equivalence Theorem 71
4. Performance Benchmark 73
5. Architecture 73
6. Development of a neural toolbox 78
7. Conclusion 79

Chapter 4. Learning Algorithms 81
1. Complex dynamics 83
2. The Learning Task 84
3. The “Out-supervised” Learning Algorithm 90
4. The “In-supervised” Learning Algorithm 93
5. Conclusion 96

Chapter 5. Capacitive and dynamical results 101
1. Encoding Capacity 101
2. Complex dynamics in Hebbian Memories 110
3. Using Complex dynamics 124
4. Conclusion 131

Chapter 6. Working Memory Model 135
1. Definitions 135
2. Cell Assemblies 136
3. Implementation 138

4. Conclusion 146

Chapter 7. Online Adaptation Process 153
1. Network Plasticity: Creation of cell assemblies 153
2. Cell assembly formation 158
3. The Retroaxonal Hypothesis 159
4. Results for Online Formation of CAs 163
5. The Context Layer 173
6. Conclusion 178

Chapter 8. Discussion 181

Appendix A. Neural Development Kit 189
1. Requirements 189
2. Implementation 190
3. Design 190
4. Neural Scripting Interface 197

Appendix B. Neural Development Interface 199

Appendix. Bibliography 203

List of Figures

2.1 Basic description of a neuron 11
2.2 Lateral view of the hippocampus 26
2.3 Coronal section of the brain of a macaque monkey 30
2.4 Hippocampal EEG and CA1 neural activity in the rats 32
2.5 Stylized nerve cell vs formal unit 36
2.6 The two main classes of neural networks 37
2.7 Decomposition of a feed forward neural network into layers 39
2.8 The Lorenz attractor 44
2.9 The Sierpinski triangle 47
2.10 The bifurcation diagram of the logistic maps 49
2.11 A flip bifurcation and a saddle node bifurcation 54
2.12 Hopf bifurcation 54
2.13 Period doubling road to chaos 56
2.14 Quasi-periodic road to chaos 57
2.15 Intermittency Chaos 58
2.16 One neuron's dynamics 59
2.17 Path-bifurcation for the one-neuron recurrent neural network 61
2.18 Path-bifurcation diagram for a 3-neurons recurrent neural network 61

3.1 Basins of attraction with their fixed-point attractors 66
3.2 Picture of a fully recurrent neural network 68
3.3 Threshold function of the rate firing model 70
3.4 Simulation performance of two models 74
3.5 Architecture of the system 75
3.6 The inhibitor activation function 78

4.1 Capacitive vs. complexity 84
4.2 Unfolding process of a 3-neuron recurrent network 85
4.3 Comparison of learning time: BTTP vs random brute force 86
4.4 Capacity vs. complexity after learning 88
4.5 Bifurcation prevents gradient descent. Example of how a bifurcation can prevent a gradient descent algorithm to converge. 88
4.6 Stability of cycles in continuous networks 94

5.1 Limited architecture of the model 102
5.2 Percentage of correct recalls with the "out-supervised" algorithm 103

5.3 Content-addressability of the hetero-associative system 104
5.4 Context dependent data set 105
5.5 Capacity of the “out-supervised” algorithm 106
5.6 Noise tolerance of the “out-supervised” learning 107
5.7 A data set with (cyclic) sequence sharing patterns 108
5.8 Performance of the two learning algorithms 109
5.9 Noise tolerance of both algorithms 111
5.10 Dynamics of the network after an “out-supervised” learning 113
5.11 Dynamical properties after “out-supervised” learning of multiple cycles 114
5.12 Dynamical properties after “out-supervised” learning of one big cycle 114
5.13 Roads to chaos through Hebbian learning in recurrent neural networks 116
5.14 The three most characteristic chaos observed after learning 118
5.15 Dynamics of the network after an “in-supervised” learning 119
5.16 Variation of the network’s dynamics 121
5.17 Lyapunov spectrum of the two learning algorithms 122
5.18 Presence of the nearby limit cycle attractors 123
5.19 Proportion of the different symbolic attractors 126
5.20 Proportion of the different types of attractors 127
5.21 Noise injection mechanism 128
5.22 How noise affects memory retrieval 129
5.23 Return maps of the chaotic dynamics and the impact of noise observed for a learned stimulus 130
5.24 Return maps of the chaotic dynamics and the impact of noise when an unlearned stimulus is fed 130

6.1 Limited architecture of the model 139
6.2 Content-addressable and working memory features 140
6.3 Noise tolerance (2 CAs) 142
6.4 Noise tolerance (5 CAs) 143
6.5 Dynamics of a CA during stimulus 144
6.6 Dynamics of the system during stimulus 145
6.7 Spontaneous dynamics of the system 146
6.8 Network dynamics for ambiguous external stimuli 147
6.9 Dynamics of a CA during stimulus for large data sets 148
6.10 Dynamics of the system during stimulus for large data sets 149
6.11 Spontaneous dynamics of the system for large data sets 150
6.12 Probability of chaos in the system 151
6.13 Probability of chaos in the system for large data sets 152

7.1 Basic steps of the Hebian/anti-Hebbian learning algorithm’s implementation 154
7.2 Upper bound of $\delta(w_{ij})$ 156
7.3 Impact on the weights of the Hebbian learning 158
7.4 The retroaxonal hypothesis 160

7.5 Basic steps of the retroaxonal learning algorithm's implementation 162
7.6 Content-addressable and working memory features after learning 165
7.7 Weight matrix after learning 166
7.8 Capacity of the working memory 167
7.9 Capacity of the working memory (ratio) 168
7.10 Highly correlated pairs of inputs present in a random data set 169
7.11 Noise tolerance 170
7.12 Dynamics of the system after learning 172
7.13 Probability of chaos after learning 173
7.14 Architecture of the system 174
7.15 Normalized activity for units belonging to a same cognitive map 175
7.16 Reactivation of CA using contextual information 176
7.17 Benefits of the context layer 176
7.18 Benefits of the context layer for ambiguous recall 177

A.1 Class diagram of the NDK 191

B.1 The NDI with the configuration manager widget. 200
B.2 The NDI during learning. 201
B.3 The NDI with some plots 202

List of Abbreviations and Symbols

Abbreviations

ART	Adaptive Resonance Theory
BPTT	Back-Propagation Through Time
CA1-4	Cornu Ammonis 1-4
CA	Cell Assembly
CM	Cognitive Map
DG	Dentate Gyrus
EC	Entorhinal Cortex
EEG	Electroencephalogram
FFT	Fast Fourier Transform
fMRI	Functional Magnetic Resonance Imaging
HM	Henry Gustav Molaison
LIA	Large Irregular Activity
LTD	Long-Term Depression
LTP	Long-Term Potentiation
MEG	Magneto Encephalogram
MLP	Multilayer Perceptron
NDI	Neural Development Interface
NDK	Neural Development Kit
NSI	Neural Scripting Interface
PCA	Principal Component Analysis
PDP	Parallel-Distributed Processing
PET	Positron Emission Tomography
RBF	Radial-Basis Function
REM	Rapid Eye Movement
RNN	Recurrent Neural Network
RRP	Readily Releasable Pool
SOM	Self-Organizing Map
TDL	Tapped-Delay-Line
TLFN	Time-Lagged Feed Forward Network

Symbols

$\mathbf{A}(t)$	The set of selected cells by the unsupervised Hebbian process at time step t.
$\mathcal{A}$	An non-chaotic attractor.
$\mathfrak{A}$	A neural network A
$\mathfrak{B}$	A neural network B
A	The associative layer.
$\mathbf{A_{fa}}(t)$	Subset of $\mathbf{A}(t)$ containing fed active cells only.
$\mathbf{A_{fi}}(t)$	Subset of $\mathbf{A}(t)$ containing fed inactive cells only.
$\mathbf{A_{ua}}(t)$	Subset of $\mathbf{A}(t)$ containing unfed active cells only.
$\mathbf{A_{ui}}(t)$	Subset of $\mathbf{A}(t)$ containing unfed inactive cells only.

α	Load parameter: the number of data in a data set over the network's size.
α_r	Improved load parameter: the number of patterns in a data set over the network's size.
$\mathcal{B}$	A basin of attraction.
β	Expected average activity of the system impinging the global inhibitor.
$\mathbf{C}$	A state vector of the context layer.
C	The context layer.
χ^{μ}	The vector representing the stimulus to learn for the data $\mathcal{D}^{\mu}$.
χ_i^{μ}	The value of the i^{th} element on the vector χ^{μ}.
χ_{i,ln_b}^{μ}	The noisy value of the i^{th} element on the vector χ^{μ}, where noise strength is ln_b.
$\chi_{ln_b}^{\mu}$	The vector representing the noisy version of the stimulus to learn for the data $\mathcal{D}^{\mu}$, where noise strength is ln_b.
$c_i\left(t\right)$	Value of the neuron i of the context layer at time t.
$\mathcal{D}$	A data set for the "out-supervised" learning algorithm.
$\mathcal{D}_{am}$	A data set for the "in-supervised" learning algorithm after mapping.
$\mathcal{D}_{bm}$	A data set for the "in-supervised" learning algorithm before mapping.
$\Delta\left(i,j\right)$	Function used to modify the weights between the units i and j in the associative layer by the unsupervised Hebbian process. Reinforce the weight in absolute values
$\delta\left(w_{ij}\right)$	Reinforcement function applied to the units i and j in the associative layer by the unsupervised Hebbian process.
$\delta_i\left(t\right)$	Error term for the neuron i at time t during back propagation with the BPTT.
$\mathcal{D}F$	The Jacobian matrix of partial derivatives of F (assuming F is of class C^1).
d_H	Hamming distance.
$d_H\left(ln\right)$	Hamming distance between the noisy pattern and the original.
$\mathcal{D}^{\mu}$	A data from the data set $\mathcal{D}$.
$\mathcal{D}_{am}^{\mu}$	A data from the data set $\mathcal{D}_{am}$.
$\mathcal{D}_{bm}^{\mu}$	A data from the data set $\mathcal{D}_{bm}$.
F	The activation function of the associative layer.
F_{ρ}	Evolution function of a dynamical system under the family of parameters ρ.
G	The activation function of the context layer.
γ	Secondary learning rate of the unsupervised Hebbian process.
$\Gamma\left(i,j\right)$	Reinforcement function applied to the units i and j in the associative layer by the unsupervised Hebbian process.
H	The activation function of the global inhibitor.
$\mathbf{I}$	The identity matrix.
IN_i	All the incoming weights to the neuron i in the associative layer.
$\iota_i\left(t\right)$	Value of the neuron i of the input layer at time t.
κ_i	Noise applied to create original parameter in the "in-supervised" algorithm.
K_i	Total activity impinging a context neuron i.
λ	Lyapunov exponent.
l_{μ}	The period of the sequence ς^{μ} in the data $\mathcal{D}^{\mu}$.
ln_s	Noise applied during learning to the external stimulus.
ln_s	Noise applied during learning to the internal states.
m_{0b}	Overlap in the external stimulus with the correct stimulus.
m_{0s}	Overlap in the internal state with the expected pattern.
$\mathtt{max}_{cs}$	Maximum authorized cycle size of the "in-supervised" learning algorithm.
$\mathtt{min}_{cs}$	Minimum authorized cycle size of the "in-supervised" learning algorithm.
m^{μ}	Overlap between the pattern μ and it's noisy version.

m_0^μ	Initial overlap between the pattern μ and it's noisy version.
μ	Parameter used in the unsupervised Hebbian process to compute the maximum allowed incoming and outgoing connection value.
$\mathcal{N}$	A neighborhood in a topological space.
ν	Learning rate of the retroaxonal learning process.
nu	Noise applied after learning to the internal states.
nu_b	Noise applied after learning to the external stimulus.
ω	Minimal weight value to respect (in absolute value) in the weakening function $\Omega\left(i,j\right)$ in the unsupervised Hebbian process.
$\Omega\left(i,j\right)$	Weakening function applied to the units i and j in the associative layer by the unsupervised Hebbian process.
OUT_i	All the outgoing weights to the neuron i in the associative layer.
p	The proportion of cells in a cell assembly overlapping with other cell assemblies.
p_e	"Expansion period" of the "in-supervised" algorithm.
p_s	"Compression period" of the "in-supervised" algorithm.
q	The proportion of cells lying in two cell assemblies.
r	The number of cells composing a cell assembly.
$\mathbf{S}$	A state vector of the input layer.
S	The input layer.
σ	Slope of the inhibitor neuron's activation function.
$\mathcal{U}$	An unstable non-chaotic attractor.
$\boldsymbol{\Upsilon}$	A state vector of the global inhibitor.
Υ	The global inhibitor.
$\Upsilon\left(t\right)$	Value of the global inhibitor at time t.
$\mathbf{V}$	The global state vector.
ε	Learning rate of the unsupervised Hebbian process.
ε_b	The learning rate of the iterative Hebbian learning algorithm applied to the weights from the input layer.
ε_s	The learning rate of the iterative Hebbian learning algorithm.
ς^μ	The vector representing the cycle, of size l_μ, associated to the data $\mathcal{D}^\mu$ and composed of the pattern $\varsigma^{\mu,k}$.
$\varsigma^{\mu,k}$	The vector representing the pattern k to learn for the data $\mathcal{D}^\mu$.
$\varsigma_i^{\mu,k}$	The value of the i^{th} element on the vector $\varsigma^{\mu,k}$.
$\varsigma_{i,ln_s}^{\mu,k}$	The noisy value of the i^{th} element on the vector $\varsigma^{\mu,k}$, where noise strength is ln_s.
$\varsigma_{ln_s}^{\mu,k}$	The vector representing the noisy version of the pattern k to learn for the data $\mathcal{D}^\mu$, where noise strength is ln_s..
$\mathbf{w_{ij}}$	
$\mathbf{W}$	The global weight matrix.
$\mathbf{w_{0j}^A}$	The weight from neuron j in the associative layer to the global inhibitor.
$\mathbf{W_A}$	The weight matrix from the associative layer to the global inhibitor.
$\mathbf{w_{ij}^C}$	The weight from neuron j in the associative layer to neuron i in the context layer.
$\mathbf{W_C}$	The weight matrix from the associative layer to the context layer.
$\mathbf{w_{i0}^I}$	The weight from the global inhibitor to neuron i in the associative layer.
$\mathbf{W_I}$	The weight matrix from the global inhibitor to the associative layer.
$\mathbf{w_{ij}^R}$	The weight from neuron j to neuron i in the associative layer.
$\mathbf{W_R}$	The weight matrix of the associative layer.
$\mathbf{w_{ij}^S}$	The weight from neuron j in the input layer to neuron i in the associative layer.
$\mathbf{W_S}$	The weight matrix from the input layer to the associative layer.

$\mathbf{w_{ij}^V}$	The weight from neuron j in the context layer to neuron i in the associative layer.
$\mathbf{W_V}$	The weight matrix from the context layer to the associative layer.
$\mathbf{X}$	A state vector of the associative layer.
$x_i\left(t\right)$	Value of the neuron i of the associative layer at time t.
$\bar{x}_i$	Average activity of the unit i in the associative layer over the last T time step (where T is a parameter of the retroaxonal learning process.
ξ^μ	The vector representing a data in a data set composed of fixed-point data element without stimulus (i.e: Hopfield networks).
ξ_i^μ	The value of the i^{th} element on the vector ξ^μ
ξ_{i,ln_s}^μ	The noisy value of the i^{th} element on the vector ξ^μ, where noise strength is ln_s.
$\xi_{ln_s}^\mu$	The vector representing a noisy data in a data set composed of fixed-point data element without stimulus (i.e: Hopfield networks), where noise strength is ln_s.

CHAPTER 1

Introduction

The brain has been the center of attention for centuries. It is believed to be the center of awareness and consciousness, since the early days. Hippocrates [400BC] supported that idea in his account of epilepsy (“On the Sacred Disease”) :

> Men ought to know that from nothing else but the brain come joys, delights, laughter and sports, and sorrows, griefs, despondency, and lamentations. . . . And by the same organ we become mad and delirious, and fears and terrors assail us, some by night, and some by day, and dreams and untimely wanderings, and cares that are not suitable, and ignorance of present circumstances, desuetude, and unskillfulness. All these things we endure from the brain, when it is not healthy. . .

However, at first sight the brain seems inert, thus Aristotle advocated that it was merely a cooling system for the blood, and that the heart was where the center of consciousness should be [Mason, 1962]. Of course, with today’s progress this idea seems asinine and the complexity of the brain never ceases to amaze.

In fact, the brain is the most complex biological machinery known to man; even with all the progress that has been accomplished in its study, it’s inner workings still remain a mystery. The 1990s were declared years of the brain because of the general resolution to decipher exhaustively the cerebral code. This has led to a expanding of neuroscience research centers. Today, the workings of individual neurons are well understood and analyzed in great details, but the way they interact and collaborate to generate high-level behavior is still very difficult to decipher. However, modern methods of observation highlighted the brain’s high-level of organization. For instance, being able to observe larger groups of cell has led to discoveries such as the place cells: a group of neurons activating simultaneously depending on spatial localization of a subject O’Keefe and Dostrovsky [1971]. Furthermore, the complexity and high dimensionality of this deciphering task has led to the creation of a new field: the computational neuroscience.

One particular aspect of the brain has been the center of the studies in the recent decades: the memory. The brain shows incredible properties when it comes down to store, process and recall information. For one, its capacity seems limitless, in time and space; moreover, the brain seems very good at working with partial information and still manages outstanding performance (i.e: recognizing the face of someone from a partial (or blurry) picture).

The study of the brain has touched many fields, such as philosophy, biology, psychology, but it has also spread in the past decade to research domains such as computer science, engineering and mathematics (with dynamical system theories). But in these fields, the research was not only about understanding the brain, but also trying to reproduce some of its wonders. As many other phenomena occurring in nature, the brain became a source of inspiration.

From the beginning of mankind, there has always been a desire to build a man or an artificial intelligence (e.g: the golem or the pygmalion myths). This idea has been modernized and the current holy grail of these researchers is to create intelligent robots.

Pascal is credited with the first realization of an "intelligent" machine, the first mechanical digital calculating machine, in 1642. Even if this is not what someone would call intelligence today, it still was a huge leap since nobody thought that cognitive human activity could be automatized. Which begs the question: how does one properly define intelligence?

Unfortunately, this question has no correct answer: while modern computers can outperform any man when it comes down to doing calculus, they fail to perform some very easy tasks for any human such as learning to speak. Nevertheless, this did not keep scientists from trying to reproduce these features. It is possible to have computers play games, do advanced mathematics (such as analytic resolution of equations, theorem proving, . . .) or even mimic human conversation. The more complex tasks machines were able to perform, the more defining "intelligence" properly seemed impossible.

In the 1950s, Alan Turing came up with a test that was supposed to help determine a machine's sentience. The basic idea behind this "Turing Test": a human judge engages in a natural language conversation with one human and one machine, all participants being placed in isolated locations. If the judge cannot reliably tell the machine from the human, the machine is said to have passed the test. Even if this may appear trivial or unhelpful, Turing underlined something very important: instead of trying to define intelligence, it is probably best to see if the system is able to output behavior that an arbitrary observer will qualify as intelligent. This is a trend that pretty much defines modern artificial intelligence, where researchers try to reproduce intelligent behavior observed in humans (and particularly in the brain).

Humans operate by manipulating high-level symbols when doing "intelligent" tasks. This observation has given birth to the traditional approach of artificial intelligence (AI, also described today as symbolic artificial intelligence). This idea is closely related to the computationalism theory, which argues that mental activity is computational. In other words, the mind operates purely by performing formal operations on symbols (much like a Turing machine). In this theory, the brain is just a biological implementation of symbolic processing machines.

In 1980, the philosopher Searle [1984] described the "Chinese room argument" experiment. This experiment requires a non-Chinese-speaking person to sit in a room, where he/she receive messages written in Chinese; after this, the subject was given a detailed response to provide, so the Chinese interrogator was convinced that the person in the room understood Chinese. Searle described this as a symbolic manipulation without having any real understanding of what those symbols mean, and claimed that it was similar to what computers do.

Searle agrees that computers will eventually be able to perform every intellectual feat humans are capable of, yet will still be lacking subjective consciousness. Searle's position has been called weak AI, and contrasts with strong AI, which claims that intelligent machines will eventually possess consciousness, self-awareness and emotions. Computationalism supports the same idea as strong AI from another perspective: instead of claiming that machines can have consciousness, it claims that the human (whom have consciousness) are (symbolic computational) machines.

Still according to Searle [1992], the computers lack the principle of causality, while "the brains cause the minds", this cannot be said about the output of a computer and its constitutive silicon chips.

The eighties put heavy criticism on the top-down approach and symbolic AI, resulting in the creation of the connectionist and bottom-up approach. Given the context in which the connectionist models have come to be, they were always considered an orthogonal approach to the computationalist models.

Based on the architecture of neurons, synapses and dendrites in the brain, the neural network models became the most famous and widely used connectionist models. Neural networks are classified in two main groups, the feed forward neural networks (where data is propagated linearly from input to output) and the recurrent neural networks (RNNs). The feed forward networks are unlikely from a biological point of view, but have produced successful practical applications. On the other hand, recurrent neural networks are biologically plausible, but still need to find a convincing application.

This thesis is in line with the current connectionist effort: gaining a better understanding of the non-linear dynamical phenomena occurring in fully recurrent neural networks, hoping to discover engineering and/or cognitivist applications. It presents a neural network model for memory-like tasks, based on a set of assumptions that, for the last twenty years, have been expressed in the fields of information processing, neurophysiology and cognitive sciences.

The first assumption states that the brain is a dynamical system and thus possesses attractors, which are the natural way to encode information. Thus, the proposed neural network model should support complex dynamics. Recurrent neural networks fit perfectly this role and can be studied as dynamical systems, which in turn guarantees the presence of attractors. From there, the model must encode information in the attractors present in this system and retrieve information by stimulating the network so as to trap its dynamics in the desired item's basin of attraction. The reasons behind the use of dynamical attractors can be found in their intrinsic stability and robustness. Since Grossberg [1992] and Hopfield [1982] precursor works, the privileged regime to code information has been fixed-point attractors.

Others have reported that the brain's dynamics are much more complex than fixed-point attractors and are more likely to be cyclic and weak chaotic regimes [Babloyantz and Lourenço, 1994; Nicolis and Tsuda, 1985; Rodriguez et al., 1999; Skarda and Freeman, 1987]. In addition to those evidences from biology, many theoretical and experimental works have shown and discussed the poor storing capacity of networks using only fixed-points attractors [see Amit et al., 1985; Domany et al., 1995, for review]. Molter and Bersini [2003a,b] have shown how networks with a randomly generated synaptic matrix allows the manipulation of a huge number of static and above all cyclic attractors for information encoding. An information is made of a pair of data: the stimulus and the corresponding limit cycle attractor. Promising results were obtained showing both a high encoding potential and a relationship between this potential and the chaotic dynamics present in the network: high potential networks have a stronger presence of chaotic dynamics.

The second quasi-unanimous view shared amongst researchers is that the learning mechanism of a neural network is somehow based on a local Hebbian mechanism. Such algorithms learn information through a supervised practice or by revealing some statistical regularities in the data through an unsupervised process. This thesis proposes various learning algorithms and studies their properties and consequences. It first covers the gradient-based learning algorithm and shows how this class of learning process is inadequate for memory-like tasks. Later three different Hebbian learning algorithms are proposed. Each algorithm tries to relax constraints existing in the previously proposed one. For example, the first algorithm is supervised and learns a priori defined data sets. The second is still supervised, but this time generates itself the output to learn and only the stimuli are a priori specified. Finally, the last algorithm features an unsupervised online learning policy to avoid the supervision in the learning phase, and also tries to be more biologically plausible, where as the other algorithms obviously are not. However, it is important to note that, even if the system is supported by some biologically unlikely hypotheses, it is possible to show that important properties of the system are invariant in regards to those hypotheses.

Even if scientists agree on the presence of dynamics in the brain and most of them admit that those dynamics are not simple fixed-point attractors, the presence of chaos (and benefits it can provide to a system) stays a much more debated hypothesis. Even though no practical application has yet seen the day, chaos plays an important role in the ideas discussed in this thesis. Nevertheless, chaos is not used as is when building memory models with recurrent neural networks. Here, chaos is more an outcome of the learning rather than a fundamental actor whose utility can be easily identified and exploited.

Since the seminal paper of Skarda and Freeman [1987], many authors share the idea that chaos is the ideal regime to store and efficiently retrieve information in neural networks [Freeman, 2002; Guillot and Dauce, 2002; Kaneko and Tsuda, 2003; Pasemann, 2002]. Theoretically speaking chaos inherently possesses an infinite amount of cyclic regimes that can be exploited for coding information. Moreover, it randomly wanders around these unstable regimes in a spontaneous way, thus rapidly proposing alternative responses to external stimuli and being able to easily switch from one of these potential attractors to another in response to any incoming stimulus. More recently, the increasing power of computers and the development of new statistical mathematics demonstrated the necessity to rely on more complex dynamics [e.g. Kenet et al., 2003].

The goal of this thesis is to take a closer look at neural networks with a brain-like memory application in mind. It aims at providing a biologically plausible model that can reproduce or mimic observations made by neuroscientists. Yet, it is not built to predict or accurately model the brain of a particular living being. The model is voluntarily kept as simple as possible in order to highlight the fundamental requirements behind different complex features of the brain. Incidentally simple models are much more computationally friendly and thus can help building larger networks within a given technological limit.

In Chapter 2, various preliminary notions are reviewed for a better understanding of this work. This chapter covers three major subjects: in the first part, a biological background is established, ranging from neurophysiology to the definition of memories. The goal is to give the reader a comfortable understanding of what is assumed about the brain and its inner working. The second part of this chapter discusses connectionism and gives a short review of this very large field. It tries to legitimate the choice of neural networks to model brain-like memory instead of a more classical computational approach. Finally, as suggested earlier, complex dynamics seem to be a very important part of the brain activity and thus naturally need to be present in a memory model. The last part of this chapter provides an introduction to the dynamical system theory and to the numerical tools used all along the thesis.

Chapter 3 starts by introducing the related approaches and comparing them to what is done here and how they can be used or related to the problem at hand. Next, it presents the model used through this thesis, in its most generic form.

Based on the results of Molter and Bersini [2003a,b] the next step was obviously to propose a learning algorithm that can achieve similar results autonomously. This thesis takes a look at classical approaches (i.e: gradient descent) even if they are not very likely from a biological point of view [Molter, Salihoglu, and Bersini, 2004a].

After that, various algorithms are studied to match the results obtained with random networks through learning. Two different learning tasks are proposed [Molter, Salihoglu, and Bersini, 2007b] where the coding of information is done in robust cyclic attractors. To follow neurophysiological observations, the learning of the synaptic matrix was based on local asymmetric Hebbian mechanisms [Bi and Poo, 1999; Levy and Steward, 1983]. These algorithms are covered in Chapter 4.

The first task consists in mapping a set of external stimuli to a set of fully specified cyclic attractors. The second task corresponds to a less supervised (and biologically more plausible) mapping: the semantics of the attractors to be associated with the feeding stimulus is left unprescribed. In this view, what is analyzed is the capacity of the network to create its own representations. Here these two types of learning are called out-supervised and in-supervised. One of the most interesting result shows that the more information is to be stored, the more chaos appears as a regime in the back, erratically itinerating among brief appearances of the learned attractors.

Chapter 5 analyzes these algorithms from different perspectives such as capacity, noise tolerance and dynamics. These tests show that chaos does not appear to be the cause, but the consequence of the learning. However, it appears as a helpful consequence that widens the network's encoding capacity. Also, chaos is not a meaningless consequence of learning: by it very nature it is easy to identify and thus can be put to good use to help the system during recall phases [Salihoglu et al., 2007]. Basically, if the output of the system is chaotic, it is clear that the system did not converge and needs help. This help can be provided with a simple noise addition to perturb the system.

Noise in dynamical systems is usually considered a nuisance. However, contrary to intuition, it has been reported that noise can have beneficial effects [Uwate and Nishio, 2005], especially in nonlinear systems driven by weak inputs. Such a positive effect of noise was first investigated by physicists and globally termed stochastic resonance [Moss et al., 2004; Wiesenfeld and Moss, 1995], in which the signal-to-noise ratio is maximal for a nonzero level of noise. Since then, many neurophysiological evidences [Kitajo et al., 2003; Russel et al., 1999], and computational models [Collins et al., 1995; Silberberg et al., 2004; van Vreeswijk and Sompolinsky, 1996] have demonstrated that noise can play a constructive functional role in brain dynamics. Accordingly, it appears important to involve stochastic noise in brain models to have a better understanding of the computational properties of internally generated brain states [Destexhe and Contreras, 2006]. In line with these results, it has been proposed that the presence of a low level of stochastic noise can result in taming chaos by stabilizing the itinerant trajectories [Freeman et al., 1997; Kozma, 2003]. Here, after learning, depending on the initial state of the network, the dynamic iterates through a chaotic trajectory. It is shown how adding stochastic noise helps in the retrieving task. It has a positive effect because it stabilizes the chaotic trajectories in the expected learned limit cycle attractor. Moreover, chaos also has implicit side effects such as its ability to fill the space and avoid proliferation of spurious data [Molter, Salihoglu, and Bersini, 2006a]. In this perspective, symbolic investigations on the dynamical attractors obtained when the network is fed with ambiguous stimuli are performed.

Even though those algorithms provide good results, they lack biological likelihood. In this regard, the proposed model needs to be improved. Even if the nature of information, and particularly memories, in the brain remains an open question, some theories seem more likely than others. In that regard, one logical step is to build a memory model based on those theories. More than fifty years ago, Hebb proposed the cell assembly theory of cortical associative memory [Hebb, 1949]. In this theory, each memory is defined by a cell assembly, i.e. a set of cells having strong synaptic weights between each others due to the well-known Hebbian rule of synaptic plasticity. The functional principles underlying that theory of memory have been formalized mathematically as attractor neural networks, and it is still today a working concept in the neuroscience community for the understanding of how the brain works. This thesis tries to conciliate these two views and proposes the encoding of information in predefined cell assemblies, here noted CA(s), characterized by complex dynamics based on a very simple rate firing model.

To validate this model as a memory, Chapter 6 tests it for two defining features: first, the ability to recover the full information from partial stimulation (content addressability), second, the ability to maintain a memory of the stimulus in the network's dynamics (working memory). The working memory appears as a fundamental component in the realization of higher cognitive functions, and defines the ability to hold and manipulate limited amounts of information during short time periods [Baddeley and Hitch, 1974]. The neural basis of the working memory has been widely investigated in primates with single cell recordings [Fuster, 1973; Fuster and Alexander, 1971; Rainer et al., 1998] and neuroimaging tools [Cohen et al., 1997]. It was demonstrated that some of the cells which were responsive to the stimulus maintained their activity during a short period after stimulus offset. In response, several computational models have already shown that cell assemblies could work as working memory by actively holding a limited amount of information for a short time (e.g: [Compte et al., 2000; Durstewitz et al., 2000; Molter et al., 2009; Mongillo et al., 2008]).

Next, as always, the model needs to be supported by a robust learning procedure. In Chapter 7, a hybrid procedure is proposed to create cell assemblies in response to external stimuli. The procedure combines two biologically plausible mechanisms. First, the rapid Hebbian/anti-Hebbian learning of the network's recurrent connections to create the cell assemblies. Second, a slow feedback mechanism to organize the incoming connections for the stabilization (or destruction) of the cell assemblies. This retroaxonal feedback has been observed on several levels in the brain [Buss et al., 2006; Hamburger, 1992, 1993; Oppenheim, 1991] and has recently been suggested as a plausible mechanism for stabilizing neuronal activity [Harris, 2008]. Results show that the obtained CAs exhibit similar behavior as the pre-encoded ones.

This algorithm is reminiscent of a long tradition of models promoting the unsupervised self-organization of information in neural networks, such as the adaptive resonance theory (e.g. [Carpenter and Grossberg, 1988; Grossberg, 1993]) or the self organizing maps (e.g: [Kohonen, 1982, 2001]). However, this model differs radically regarding the nature of the expected dynamics. While in the former models the successful encoding/retrieval of information was characterized by simple dynamics (usually in the form of fixed point attractors), here, following the idea that the presence of chaotic dynamics can boost the network's capacity [Molter and Bersini, 2003a,b], complex dynamics were enforced as much as possible.

Finally, this model is proposed as a working paradigm for the formation of a cognitive map. In that view, a map results from the juxtaposition of several cell assemblies associated with the environment's set of stimuli. After learning multiple maps, a context layer is used to help recover the map a stimulus belongs to. In return, the context knowledge (from the previous stimuli or top-down control) can help identify noisy stimuli and can precisely help to disambiguate external stimuli, which could be associated with CAs from multiple maps. This model of cognitive map formation based on the creation of cell assemblies is an alternative to the view that a cognitive map is characterized by a continuous attractor [McNaughton et al., 2006; Samsonovich and McNaughton, 1997].

Publications

International Journals.

- **Neural Networks - Special Issue Invited, 2009:**
  "Online unsupervised formation of cell assemblies for the encoding of multiple cognitive maps"
  Salihoglu, Bersini, Yamaguchi, and Molter
- **Neural Computation, 2007:**
  "The road to chaos by hebbian learning in recurrent neural networks"
  Molter, Salihoglu, and Bersini

Chapter in Book.

- **Neurodynamics of Cognition and Consciousness, 2007:**
  "Giving meaning to cycles to go beyond the limitations of fixed point attractors"
  Molter, Salihoglu, and Bersini
  Springer-Verlag Berlin and Heidelberg GmbH & Co. K (ISBN-13: 978-3540732662)

International Conference Presentations with proceedings.

- **Proceedings of the IJCNN conference, 2009:**
  "Online organization of chaotic cell assemblies. A Model for the Cognitive Map Formation?"
  Salihoglu, Bersini, Yamaguchi, and Molter
  [Runner-Up Student Paper]
- **Proceedings of the ISNN conference, 2008:**
  "Visual scene identification through complex dynamics in recurrent neural networks to solve the binding problem"
  Salihoglu, Molter, and Bersini
- **Proceedings of the IJCNN conference, 2007:**
  "How stochastic noise helps memory retrieval in a chaotic brain"
  Molter, Salihoglu, and Bersini

- **Proceedings of the ICONIP conference, 2006:**
  "How reward can induce reverse replay of behavioral sequences in the hippocampus"
  Molter, Sato, Salihoglu, and Yamaguchi
  [Best Paper]
- **Proceedings of the IJCNN conference, 2006:**
  "How to prevent spurious data in a chaotic brain"
  Molter, Salihoglu, and Bersini
- **Proceedings of the Nolta conference, 2005:**
  "An interpretative recurrent neural network to improve pattern storing capabilities - dynamical considerations"
  Molter, Salihoglu, and Bersini
- **Workshop of the IJCNN conference, 2005:**
  "Phase synchronization and chaotic dynamics in Hebbian learned artificial recurrent neural networks"
  Molter, Salihoglu, and Bersini
- **Proceedings of the IJCNN conference, 2005:**
  "Introduction of an hebbian unsupervised learning algorithm to boost the encoding capacity of hopfield networks"
  Molter, Salihoglu, and Bersini
- **Proceedings of the IJCNN conference, 2005:**
  "Learning cycles brings chaos in continuous hopfield networks"
  Molter, Salihoglu, and Bersini
  [Best Student Poster Paper]
- **Proceedings of the IJCNN conference, 2004:**
  "How chaos boosts the encoding capacity of small recurrent neural networks: learning consideration"
  Molter, Salihoglu, and Bersini

Technical Reports.

- **Technical Report, IRIDIA - ULB, 2005:**
  "Storing static and cyclic patterns in an hopfield neural network"
  Molter, Salihoglu, and Bersini
- **Technical Report, IRIDIA - ULB, 2004:**
  "Hetero-associative symbolic learning using neuromodules"
  Molter, Salihoglu, and Bersini

DEA Thesis.

- **Université Libre de Bruxelles, 2004:**
  "Chaos in small Recurrent Neural Networks: theoretical and practical studies"
  Salihoglu

CHAPTER 2

Background

The work covered in this thesis is at the intersection of several fields. This chapter introduces them and establishes a sufficient background for the reader, as well as setting a common vocabulary. The first section covers the roots of this work: the neuro-physiological and neuro-dynamical observations. The second section looks at the cognitive theory of the brain. More interest is given to the memory formation, storage and processing in the third section. The fourth section introduces the hippocampus, which is a region of the brain very closely involved in memory formation. The fifth section presents computational models of biological neural networks, ranging from simple models (inspired by biological facts and evidences) to complex models that try to be accurate and predictive of the real biological systems they represent. Eventually, this overview legitimates the neural network model developed in this thesis. The last section introduces nonlinear dynamical systems, its theory and various mathematical tools used to analyze their dynamical properties. As shown in this work, complex dynamics are an important part of neural systems and seem favorable to computational models in common memory related tasks (i.e: improving the capacity of the system to: store information, correctly recall them, ...). Finally, the last section also defines and describes the tools used to analyze the dynamical properties of such neural networks.

1. Neurophysiology

1.1. The Brain. All vertebrate, and most invertebrate, animals are characterized by the presence of a central nervous system, the brain. This part of the anatomy is extremely complex, even in simple animals, yet alone the human. The cerebral cortex of the human brain contains billions of neurons, depending on gender and age [Pelvig et al., 2008], linked with up to 10,000 synaptic connections each. Each cubic millimeter of cerebral cortex contains roughly one billion synapses [Alonso-Nanclares et al., 2008].

From a philosophical point of view, it seems that the most important function of the brain is to serve as the physical structure underlying the mind. From a biological point of view, though, the most important function is to generate behaviors that promote the welfare of an animal. Brains control behavior either by activating muscles, or by causing secretion of chemicals such as hormones. Even single-celled organisms are capable of extracting information from the environment and acting in response to it [Gehring, 2005]. For example, sponges, which lack a central nervous system, are capable of coordinated body contractions and even locomotion [Nickel et al., 2002]. In vertebrates, the spinal cord by itself contains the neural circuitry capable of generating reflex responses as well as simple motor patterns (such as swimming or walking [Grillner and Wallén, 2002]). However, sophisticated control of behavior, based on complex sensory input, requires the information-integrating capabilities of a centralized brain.

Despite rapid scientific progress, much about how brains work remains a mystery. The scientific community now understands the operations of individual neurons and synapses in considerable details, but the way they cooperate in ensembles of thousands or millions has been very difficult to decipher. Methods of observation such as electroencephalogram (EEG) recording, and functional brain imaging shows that brain operations are highly organized, but these methods do not have the resolution to reveal the activity of individual neurons [van Hemmen and Sejnowski, 2005].

The brain is composed of two cell classes, the glia and the neurons. Even if there are roughly the same amount of each cell's type in the brain, the reason neurons get more attention comes from their ability to send signals to each other over long distances [Kandel et al., 2000]. This occurs through a thin protoplasmic fiber, called axon, extending from the cell body.

The axon projects itself, usually with numerous branches, to other areas, which may be nearby or in distant parts of the brain (or body). The signals transmitted by the axons are called "action potentials" and appear as electrochemical pulses. It lasts less than a thousandth of a second and travels at speeds of 1 to 100 meters per second along the axon. Some neurons emit action potentials constantly, usually in irregular temporal patterns (at rates of 10-100Hz); while other neurons are quiet most of the time, but occasionally emit a burst of action potentials.

These signals are transmitted to other neurons (or to non-neuronal cells) by means of specialized junctions called "synapses" [Kandel et al., 2000]. A single axon may make as many as several thousand synaptic connections. When the action potential reaches the synapse, it causes the release of neurotransmitter from the synaptic vesicles. The neurotransmitter binds to receptor molecules in the target cell's membrane. There are different neuronal receptor types: excitatory, meaning they increase the rate of action potentials in the target cell; inhibitory, meaning they decrease the rate of action potentials; or have complex modulatory effects on the target cell.

It is important to note that the brain is not an all-electrical continuous network, it needs biochemical changes to propagate signals. This is a key element of learning that exists in the brain. It also explains how by modifying biochemical reactions, different factors like drug or alcohol consumption can influence the perceptions. Psychotropic drugs often act precisely in the same way as these neurotransmitters.

1.2. The Nerve Cell: The Neuron. The neuron is an excitable cell of the nervous system, whose role is to receive, to process and to transmit information. They are a core component of the brain, the vertebrate spinal cord, the invertebrate ventral nerve cord, and the peripheral nerves. However, not all neurons serve the same purpose; they are usually classified according to their function. There are sensory neurons[1], the motor neurons[2], the interneurons[3].

The sensory neurons respond to stimuli, affecting cells of the sensory organs (i.e.: touch, sound, light, ...), and carry those nerve impulses from receptors (or sense organs) towards the central nervous system. Motor neurons are usually located in the central nervous system and project their axons outside the central nervous system. They receive signals from the brain and spinal cord, and they directly, or indirectly, control muscles contractions and affect glands. Interneurons

[1]The sensory neurons are also known as the afferent neurons (or afferent nerves).

[2]The motor neurons are also known as the efferent neurons or efferent nerves.

[3]The interneurons are also called "intrinsic neurons", relay neurons, association neurons or local circuit neurons

are multi-polar neurons which connect afferent neurons and efferent neurons in neural pathways within the same region of the brain or spinal cord. The interneurons can also have connections with other interneurons to form a complex network (e.g. in the CA3 layer of the hippocampus). Like motor neurons, they are always located in the central nervous system and form the largest group in the nervous system.

Sensory neurons respond to stimuli, and communicate the presence of stimuli to the central nervous system. The central nervous system processes those information and if necessary sends responses to other parts of the body for action. Neurons do not go through mitosis, and usually the brain cannot replace them after their destruction; however astrocytes[4] have been observed to turn into neurons, as they are sometimes pluripotent.

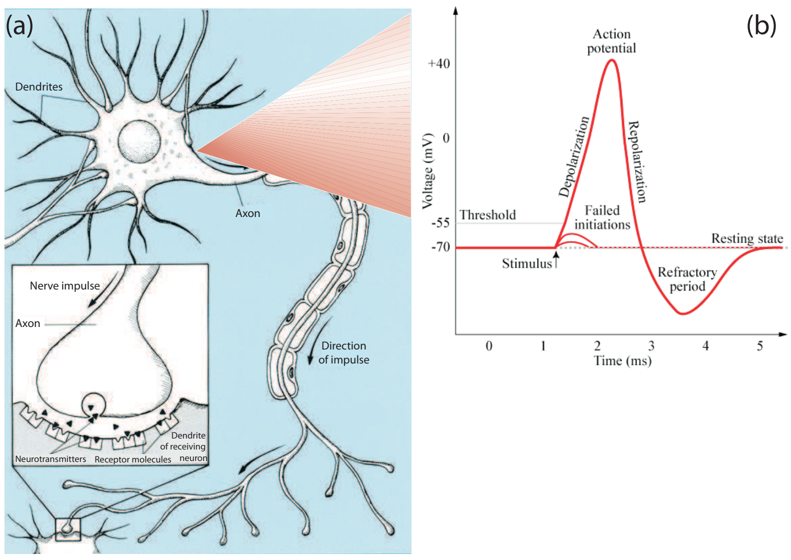


FIGURE 2.1. (a) Basic description of a neuron, its action potential and the neurotransmitters involved in the transmission of the nerve impulse to other nerves or muscles. (b) "Schematic" action potential in a neuron. Below a certain threshold the neuron fails to fire, above that threshold through a depolarization a spike occurs. With repolarization of the membrane, the spike ends and is followed by a refractory period where no other action potential can occur.

Even though neurons can have a great diversity in their function in the nervous system, they all are highly specialized for the processing and transmission of cellular signals. They can have a wide variety in their shape, size, and electrochemical properties but four morphologically defined subparts are always present (as shown in Figure 2.1):

(1) The soma[5] is the central part of the neuron. It contains the nucleus[6] of the cell, and is where most protein synthesis occurs.
(2) The dendrites of a neuron are short cellular extensions with many branches (metaphorically this overall structure is referred to as a dendritic tree). This is where the majority of input to the neuron occurs.

[4]Astrocytes are a sub-type of glial cells [Kolb and Whishaw, 2008]
[5]The soma is also known as the cell body.
[6]The neuron's main genetic information is found in the nucleus.

(3) The axon is a finer, cable-like projection, which can extend up to tens of thousands of times the diameter of the soma in length. The axon carries nerve signals away from the soma (and carries some types of information back to it). Many neurons have only one axon, but this axon may - and usually will - undergo extensive branching, enabling communication with many target cells. The tips of the axon, called “presynaptic terminals”, come into close contact with the dendrites of other neurons or with muscles.

(4) The axon terminal contains synapses, specialized structures where neurotransmitter chemicals are released in order to communicate with target neurons.

1.3. Development of the Brain. The brain does not simply grow; it develops in an intricately orchestrated sequence of steps. Many neurons are created in special zones that contain stem cells, and then migrate through the tissue to reach their ultimate location. Once a neuron is in place, it begins to extend dendrites and an axon into the area around it. Axons, because they commonly extend a great distance from the cell body and need to make contact with specific targets, grow in a particularly complex way. The axon is attracted or repelled by various cellular elements, and thus is oriented in a particular direction at each point along its path. Considering the entire brain, many thousands genes give rise to proteins that influence axonal path finding. However, the genes only partly determine the synaptic network that finally emerges. In many parts of the brain, axons initially ”overgrow”, and are then “pruned” by mechanisms depending on neural activity [Purves and Lichtman, 1985].

In humans and many other mammals, new neurons are created mainly before birth, and the infant brain actually contains substantially more neurons than the adult brain. There are, however, a few areas where new neurons continue to be generated throughout life. The two areas where this is well established are the olfactory bulb, involved in the sense of smell, and the dentate gyrus of the hippocampus, where there is evidence that the new neurons play a role in storing newly acquired memories. With these exceptions, however, the set of neurons that is present in early childhood is invariant through life. (Glial cells are different: as with most types of cells in the body, these are generated throughout the lifespan.)

Although the pool of neurons is largely in place by birth, their axonal connections continue to develop for a long time afterward. In humans, full myelination[7] is not completed until adolescence [Paus et al., 2001].

There has long been debates about whether heredity or upbringing was responsible for the qualities of mind, personality, and intelligence. The nature versus nurture debate [Ridley, 2004] is not just a philosophical question: it has great practical relevance to parents and educators. Although many details remain to be settled, neuroscience clearly shows that both factors are essential. Genes determine the general form of the brain, and how the brain reacts to experience. Experience, however, is required to refine the matrix of synaptic connections. In some respects, the development of the brain is mainly a matter of presence or absence of experience during critical periods of development [Wiesel, 1982]. In other respects, the quantity and quality of experience may be more relevant: for example, there is substantial evidence that animals raised in enriched environments have thicker

[7]Myelination helps prevent the electrical current from leaving the axon by forming a myelin layer around axons. It is essential for proper functioning of the nervous system.

cortices (indicating a higher density of synaptic connections) than animals whose levels of stimulation are restricted [van Praag et al., 2000].

At some point, authors thought that the genome provided the whole information contained in the brain. For example, Gazzaniga [1992] said:

> In the original Sperry view of the nervous system, brain and body were developed under tight genetic control. The specificity was accomplished by the genes' setting up chemical gradients, which allowed for the point-to-point connections of the nervous system.

However, if each pair within the human genome could encode 1 bit of information, this will still only represent $3.5\,10^9$bits, which is way below the complexity of the neural system present in the brain. On this basis, some neural and molecular scientists have concluded that genes could not possibly have enough storage capacity to specify all of these connections, their location and the type of neuron required (and information for the rest of the body). As Changeux [1985] noted:

> It seems difficult to imagine a differential distribution of genetic material from a single nucleus to each of these tens of thousands of synapses unless we conjure up a mysterious "demon" who selectively channels this material to each synapse according to a pre established code! The differential expression of genes cannot alone explain the extreme diversity and specificity of connections between neurons.

Aside from the theoretical considerations, solid empirical evidence comes from experiments showing considerable variations in synapse configurations among identical clones growing in the same environment.

Through experience, short and long term changes happen in the brain. Those changes, in the function and structure of the nervous system, are the natural adaptation of the brain in response to its environment. To explain this learning process different forms of neural adaptation have been proposed. These neuronal changes are most likely the physical basis of learning and memory. A typical classification of these changes is relative to their persistence.

As explained above, the most persistent and major changes occur in the early ages (up till childhood). During this period, there is a proliferation of nerve cells and high level of organization through neural pathways formation. Environmental interactions play a major part in this process and hence enable each individual's nervous system to be optimized for its own body's geometry (e.g. the distance apart of the eyes clearly affects the way the brain shapes itself [Bland, 1998]). Once this process is complete, the nervous system remains nearly identical for the rest of its lifetime. Changeux [1985] states that waves of synaptic growth occur; with subsequent experiences serving to retain the useful, while eliminating the useless and redundant ones. This hypothesis has inspired many evolutionary algorithms in connectionist models often relying on some sort of randomness.

Other changes in the brain take the form of long and short-term adaptations. Long-term adaptations are long lasting changes, which can range from mere hours to years. This occurs by the means of synaptic efficiency modification[8]. It is commonly accepted that there is no structural neural modifications at this stage. Next part discusses more in details the long-term potentiation mechanism which

[8]Synaptic efficiency is also known as synaptic plasticity

has been identified as responsible for this neural adaptation and which has given rise to a family of learning algorithms in the connectionist models, known as the Hebbian algorithms [Hebb, 1949].

Short-term adaptations reflect context modifications through fast behavioral changes. The underlying hypothesis is that complex dynamics reflect the changes and not synaptic changes. This allows the system to have strong behavioral changes that will not last nor leave any trace until a long-term adaptation kicks in. This neural adaptation based on dynamical processes is also called "dynamical adaptation" [Guillot and Dauce, 2002]. A given neural group possesses different dynamical attractors and goes from one to another based on the context.

1.4. Synaptic Plasticity. The pioneering investigations on the microscopic structure of the brain, due to the Spaniard Ramón y Cajal [1894], suggest that learning is not exclusively the product of new cell growth. In the late 19th, he suggested that memories may be formed because of connection strengthening between existing neurons (to improve their communication effectiveness). Later, Donald Hebb introduced, what is now called, the "Hebbian theory". In 1949, reminiscent of Ramon y Cajal's suggestion, he proposed that new connections could grow between cells to further enhance their ability to communicate. Hebb [1949] said:

> Let us assume then that the persistence or repetition of a reverberatory activity (or "trace") tends to induce lasting cellular changes that add to its stability. The assumption can be precisely stated as follows: When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.

This kind of neuronal plasticity is generally seen as a long lasting change in synaptic strength (i.e. the facility for a pre-synaptic neuron to influence the potential of a post-synaptic neuron). This was later confirmed experimentally by Bliss and Lomo [1973] with the discovery of the long-term potentiation (LTP) effect in the hippocampus of rats. During the experiment, the authors have done brief tetanic[9] electrical stimulation of an afferent pathway and observed an increase in synaptic efficiency (lasting from hours to days). This proved that synaptic plasticity could play a role in learning.

LTP synaptic changes have been observed in many other brain areas [Cooke and Bliss, 2006]; however the hippocampus is still a particularly favorable [Malenka and Bear, 2004] site for studying LTP because of its densely packed and sharply defined layers of neurons. The best-studied form of LTP occurs at synapses that terminate on dendritic spines and use the glutamate transmitter.

The Hebbian rule explained how mutual excitation of two cells strengthens their connections. Hebb [1949] said:

> The general idea is an old one, that any two cells or systems of cells that are repeatedly active at the same time will tend to become 'associated', so that activity in one facilitates activity in the other.

The next step for this Hebbian rule is to account for the fact that memories can be forgotten. This is suggested to happen through weakening of synaptic strength

[9]High-frequency.

between two neurons when they are not active at the same time. Hebb [1949] also has formulated this[10] :

> If the inputs to a system cause the same pattern of activity to occur repeatedly, the set of active elements constituting that pattern will become increasingly strongly interassociated. That is, each element will tend to turn on every other element and (with negative weights) to turn off the elements that do not form part of the pattern. To put it another way, the pattern as a whole will become 'auto-associated'. We may call a learned (auto-associated) pattern an engram.

This mechanism is caused by a long-term depression (LTD, opposite of LTP) [Kirkwood and Bear, 1994]. When nerve impulses reach the synapses at very low frequency, they cause an LTD. Instead of becoming more efficient, the synaptic connections are weakened.

The occurrence of these two mechanisms, amongst most of the synapses, has lead to the assumption that they play an important role in the formation and disappearance of certain types of memories. They are the most widely studied cellular models of synaptic plasticity. A better understanding of these mechanisms can lead to a better understanding of learning processes and memory formation. But this does not stop with neuroscience, since those mechanisms have made their way in the connectionist models as well, with the well known Hebbian synaptic plasticity[11], which has been the source of many algorithms.

2. The Physiological Basis of Behavior

The brain is a complex machinery which works simultaneously at several hierarchical levels. Neuroscientists try to understand which level is relevant to produce a given intelligent behavior. They have been able to locate various levels of this hierarchy as a physiological basis for behavior.

2.1. Neurons and the Connectionist approach. The first approach has its root in the work of Hubel and Wiesel [1962]. They have observed that certain cells in the visual cortex respond to particular features presented at specific locations in the visual field: they have found that various cells responded to simple features (i.e. local edge having a particular location and orientation) while others had more complex responses. This view later became known as the neural doctrine [Barlow, 1972], since it places the physiological basis of behavior at the level of individual neurons. However, there are also lower levels to look at. One such level is the biochemical changes at the synapse as the biological basis of behavior. Hameroff and Penrose [1996] locate it at the level of microtubules (nano-sized tubules running through neurons) which act as part of a quantum computer.

Later, authors such as Amari [1983]; Elman [1991]; Hopfield [1982]; Rosenblatt [1962]; Rumelhart et al. [1986a,b]; Sejnowski and Rosenberg [1987] have suggested that even though changes involved in learning behavior produce cellular and molecular modifications, to be fully understood, these processes need to be studied at a higher level. They started looking at it through the neural network layer, which

[10]This quotation is important and will be used again on the part about memories and is a building block of the main contribution of this thesis.

[11]Discussed in detail in the next section.

led to the development of basic brain's model composed of inter-connected artificial neurons, where a weight between two units represents the strength of the connections. This was later called the "connectionist approach".

The connectionist approach can be further specialized in two major categories: the feed forward networks, lacking any kind of recurrent connection in the system, act as combinatory machines, and the self-organizing dynamical systems based on recurrent neural networks. However, the foundation and the explanations of these neural networks remain at the level of the action potential of single neurons.

2.2. Neurodynamics. The knowledge and understanding of the brain has been mostly anatomical. While studies have proven, through brain lesion observations, that specific regions of the brain are associated to certain particular functions, these studies struggle to provide any conclusive information on the relation between those regions.

The idea of studying the dynamics behind the brain is not new. Neurodynamics[12] is an area of research at the borderline between neuroscience and nonlinear dynamics, complexity theory and statistical physics. It is part of the cognitive sciences, which focuses on the spatio-temporal characteristics of neural activity. It uses complex system theory to study those dynamics and describe brain function. It contrasts with the computational and modular approach of cognitive neurosciences, which puts its focus on the study of biological substrates underlying cognition, particularly the neural substrates of mental processes and their behavioral manifestations.

Perceptions have changed with the development of new tools to study the brain. Tools, such as the multi-electrode electroencephalogram (EEG) and magneto encephalogram (MEG), functional magnetic resonance imaging (fMRI) and scanning by positron emission tomography (PET-scanners), have allowed dynamic observations of the brain activity. These observations gave a look at the basis of behavior from a dynamical system point of view.

As Freeman [2002] said:

> Now, in the 21st century, the EEG will lead us in a remarkably different direction of growth for the computing industry compared to the one provided by action potentials of single neurons.

And earlier Skarda and Freeman [1990a] said:

> We agree with Searle that "[pains] and other mental phenomena just are features of the brain and perhaps the rest of the central nervous system", and that the important requirement for understanding this relationship is the distinction between micro- and macrolevels of neural functioning, Our research has led us to break with a foundational concept of contemporary research on the nervous system, the "neuron doctrine", that we and the majority of our colleagues once accepted, but which we now see as mistaken and as a source of misunderstanding in attempts to comprehend the brain as the organ of behavior.

From a dynamical point of view, the output of a single neuron is too noisy and unpredictable to be of any use, while the system as a whole is consistent and produces a coherent pattern that can be associated to the stimulus which is at the

[12]The term Neurodynamics date back to the 1940s [Burrow, 1943].

source of it. Skarda and Freeman [1990a] made one major experiment using EEG to support this dynamical paradigm. The experiment can be described as follows. They analyzed the olfactory bulb of the rabbit by simultaneously recording data from different cortical areas during cognitive tasks with the use of EEG. Their observations showed the existence of spatially organized patterns distributed across the entire bulb in response to reinforced odorant, as well as strong presence of chaotic dynamics. This chaotic dynamics seems to be the regime the brain goes to during attentive waiting states. However, when a known odor is presented, the dynamics shifts (through a bifurcation) to an almost cyclic dynamics.

Even though these results are local to the olfactory bulb, they still highlight the need to take a closer look on the brain's output with the tools and results of dynamical system theory. It has been clear since those results, that understanding the global dynamical behavior of the brain is mandatory to understand the brain itself. This has been called the dynamical hypothesis, for which a detailed introduction can be found in [Guillot and Dauce, 2002].

However, the functional significance of those results remains a matter of debate. Again according to Freeman [2000]:

> Most neuroscientists of the "neuron doctrine" reject EEG and MEG evidences, in the beliefs that recording wave activity is equivalent to observing an engine with a stethoscope or a computer with a galvanometer while the real work of brains is done by action potentials.

Two important dynamical observations are very important; the first one is the phenomenon of synchronization occurring between neuronal groups and the second one is the presence of chaotic dynamics.

2.2.1. *Synchronization of neuronal groups.* When dealing with signals, two signals are synchronous if there is a relation between their phases and frequencies. Since neurophysiology can describe neural activity as such signals, this definition can be used here as well. Phase synchrony, also just called "synchrony", indicates that a neural group oscillates at a given frequency band with precise phase-locking over a limited period of time [Nunez, 1981]. The common frequency bands are identified as delta ($[1, 4]$Hz), theta ($[4, 8]$ Hz), alpha ($[8, 12]$ Hz), beta ($[12, 30]$ Hz), and gamma ($[30, 80]$ Hz).

Gray et al. [1989]; Neuenschwander et al. [1996]; Skarda and Freeman [1987] have done experiments on animals showing strong correlation between behavioral states and transient periods of synchronization of neuronal groups in the gamma range. Later, Rodriguez et al. [1999] obtained similar results in human beings during visual cognitive tasks, while facing ambiguous visual stimuli. It appears from their research that synchrony seems responsible for the binding of different, but related, visual features so that the visual pattern can be recognized as a whole. Others have proposed these periods of synchronization as a central mechanism to integrate widely distributed neurons together into a coherent ensemble that reflects a given cognitive act. In the same vein, Skarda and Freeman [1990b]; Varela [1995] suggested that desynchronization could reflect a process of active uncoupling so that the brain can go from one cognitive state to another.

2.2.2. *Chaos in the brain.* EEG analyses of different regions of the brain allowed looking at the different dynamics the brain goes through, depending on the state the subject is in. For example, deep anesthesia, coma, or brain death show a dynamics which is an equilibrium steady state attractor, while epilepsy reflects as an abnormal phase synchrony among the entire brain, indicated by the presence of a limit cycle attractor [Cohen et al., 2002]. During awake activity with no external stimulus, the brain goes typically through an aperiodic unpredictable signal. Many authors, such as Babloyantz and Destexhe [1986]; Nicolis and Tsuda [1985]; Skarda and Freeman [1987], conclude that locally the brain background activity is a deterministic chaotic dynamics. Deeper chaotic dynamics can appear globally (at the brain scale) but this is often a sign of a pathological state (i.e. depressive state [Thomasson et al., 2000]). During cognitive tasks (or when recognized stimuli are presented to the brain), the brain's dynamics usually shift to a limit cycle dynamics.

Following these observations, Sompolinsky et al. [1988] suggested the integration of chaotic dynamics into connectionist models (see section 5 on page 33 for more details on these models) which implies that chaos may provide the basis for flexibility, adaptiveness, and trial–and–error coping that enables the nervous system's interaction with an unpredictable and ever–changing environment.

Even if EEG measures, coupled with tools from the theory of nonlinear dynamical systems, show chaotic dynamics on dendritic synaptic potentials and axonal action potentials, according to Rapp [1993], these results should be taken with caution since they can be misled by the presence of pure noise. However, considering chaotic dynamics as the basal state of behavior, and limit cycle attractors as the signature of cognitive processes sounds appealing. Still, as suggested by Skarda and Freeman [1990a]:

> It is likely that there will remain substantial uncertainty for some years, possibly decades, about the differences between a limit cycle trajectory that aborts prior to convergence to a periodic attractor, a limit cycle attractor under perturbation by noise, and a narrow spectral band chaotic attractor in which the unpredictability appears in variation of phase or in frequency narrowly about a mean.

Others [Skarda and Freeman, 1990b; Tsuda, 2001] suggested the possibility that the brain processes information exclusively through chaotic dynamics. According to this hypothesis, a cognitive process would reflect the passing from one chaotic attractor to another. One consequence of this would be on memories, since they would no longer be static elements of the brain, which can be retrieved and perfectly recovered (as with fixed point or limit cycle attractor). Here, memory recovery is an active process in which the brain re-creates it instead of recalling it. This of course implies that memory can be altered in this process.

3. Memory

Artificial neural networks are very versatile tools; they are suited for a wide array of tasks, but this thesis will look mainly at them from the perspective of information encoding. When it comes to information encoding, recalling and processing, the best known working model is the brain. The brain holds huge amount of information ranging from very volatile memories to long lasting ones. It is still not clear how the brain manages all its amazing feats, but the inner working of the brain's storing capability is a big inspiration when it comes to modeling information storing devices. It is therefore important to better understand it, in order to provide more insightful neural network models that mimic some of its features. This section tries to clarify what hides behind a generic term such as memory.

From the psychological point of view, memory is the organism's mental ability to store, retain and recall information. Traditional studies of memory began in the fields of philosophy, and later cognitive psychology. More recently, it has become one of the main pillars of a branch of cognitive neuroscience.

Considering a memory as a piece of information, three main stages are required for its formation and retrieval. There is the *encoding*[13]: receiving, processing and combining the information; the *storage*: creation of a permanent record of the encoded information; and the *retrieval*[14]: calling back the stored information in response to some cue for use in a process or activity.

Memory can be roughly divided into three main categories:

- **sensory memory** is the ability to retain impressions of sensory information after the original stimulus has ceased.
- **short-term memory** is the capacity for holding a small amount of information, in an active, readily available state, for a short time period.
- **long-term memory** is a memory that can last from a few days to several decades.

3.1. Sensory Memory. The initial 200 - 500 milliseconds after an item is perceived correspond to sensory memory. It refers to items detected by the sensory receptors which are retained temporarily in the sensory registers. It has a large capacity for unprocessed information but is only able to hold accurate images of sensory information shortly. A simple example is the ability to look at an item for a very short time and remember what it looked like. Two types of sensory memory have been particularly well studied, the visual sensory memory[15] and the auditory sensory memory[16].

Earliest studies go back to 1740, when the German physicist and mathematician Johann Andreas Segner conducted an experiment in which he attached a glowing coal to a cartwheel and rotated the wheel at increasing speeds. Segner observed that when the glowing coal completed a full rotation in less than 100ms the subjects perceived only an unbroken circle of light.

Sperling [1960] was the first one to systematically study this effect. In his experiments, the subjects were presented a grid of 12 letters over three rows during a very short (50ms) period of time (a review for this can also be found in [Baddeley, 1999]). For example:

$$\begin{matrix} P & Y & F & G \\ V & J & S & A \\ D & H & B & U \end{matrix}$$

The first set of experiments, called "whole recall", asked the participants to recall as many letters as they could. The second one, called "partial recall", repeated the same exercise but with one additional information, before the letters were shown the participants were told which row's letters they would have to recall. Results show that while subjects could recall about four letters during a "whole recall" process, they were also able to recall an average of three of the four letters of the given row in the partial recall process, even though they were told which row to focus on before the whole letter grid was presented. This indicated that for a brief time period the whole grid was available as a sensory memory to the subjects.

[13]the encoding is also known as registration.

[14]the retrieval is also known as recall.

[15]The visual sensory memory is also known as iconic memory.

[16]The auditory sensory memory also known as echoic memory.

Sperling was able to show the capacity of sensory memory to be approximately 12 items, but that it degraded very quickly (within a few hundred milliseconds). This degradation explains the inability to recall all the letters during the whole report procedure, even though the whole grid is present as a sensory memory.

Sensory memory is very short lived and operates within a time frame of a second. It is also characterized by being outside of conscious control (i.e. it happens automatically and unbidden). Another important note about sensory memory is its inability to be prolonged via rehearsal.

3.2. Working Memory. Despite retaining information for a very short period of time, sensory memory is not to be confused with working memory[17]. It allows recall for a period of several seconds to a minute (without rehearsal) of information in an active, readily available state. However, working memory has a finite (limited) capacity, also called "memory span". To test this memory span, the subjects are presented with lists of items (e.g. digits or words) of increasing length. A participant's span is determined as the longest list length that he or she can recall correctly in the given order on at least half of all trials. Miller [1956] conducted an experiments showing that the store of working memory was 7 ± 2 items[18].

Recent studies have shown this number to be accurate for college students recalling lists of digits, but memory span varies widely with the populations tested and with material used. One example is word recalling, this exercise depends on several characteristics of the words. Fewer words can be recalled when the words have longer spoken duration (known as the word-length effect) [Baddeley et al., 1975]. This is also the case when words' speech sounds are similar to each other (known as the phonological similarity effect) [Conrad, 1964]. On the other hand, more words can be recalled when the words are highly familiar or occur frequently in the language [Poirier and Saint-Aubin, 1996]. When the words come from a single semantic category (such as sports) this also increases the performance of the recall [Poirier and Saint-Aubin, 1995]. According to the available evidence, the best overall estimate of working memory is about four pieces or chunks of information [Cowan, 2001]. Most works on working memory have used verbal material, while recent researches also covered visual working memory [Luck and Vogel, 1997], and spatial working memory [Parmentier et al., 2005].

Even though the working memory has a very limited capacity, it is known it can be increased through a process called "chunking". For example, given the string:

FBIPHDTWAIBM

people are only able to remember a few letters; when the same information is presented in the following way:

FBI PHD TWA IBM

people can remember a great deal more letters. This is a consequence of the brain's ability to chunk the information in meaningful groups of letters. Even if those groups are meaningful, Simon [1974] showed that the ideal size for chunking letters and numbers was three[19].

[17]The working memory is also known as primary memory or active memory.

[18]Which is also the title of his famous paper: "The Magical Number Seven, Plus or Minus Two".

[19]In some countries, this may be reflected by the tendency to write and remember telephone numbers by dividing them into chunks of three number (with the final four-number groups generally broken down into two groups of two).

In the 1960s, it was assumed that all memories pass from working to long-term store after a small period of time. This model of the memory is known as the "modal model" and has been made famous by Atkinson and Shiffrin [1968]. This model led to lots of controversies, i.e. whether all or only some memories are retained permanently or even a more fundamental questioning on the genuine distinction between the two stores.

Anterograde amnesia[20] is one evidence in favor of the existence of a separate working memory store. Patients suffering from this kind of amnesia have shown no degradation in their ability to retain and recall small amounts of information over short period of time (up to 30 seconds). However, their ability to form longer-term memories show a dramatic impairment[21] The goal of those experiments is to show that amnesia spares the working store while long-term memory store clearly shows a degradation.

Other studies support the evidence of two separate stores by disturbing one store while leaving the other one unaffected by the process. Davelaar et al. [2005] have shown that some manipulations (e.g., a distractor task[22]) impair memory for the 3 to 5 most recently learned words of a list (presumably still held in working memory), while recall from words learned earlier (presumably stored in long-term memory) were unaffected. Other manipulations (e.g., semantic similarity of the words) seem only to affect recall from earlier list words, but recall of recent words in a list seems unaffected. These results indicate the two stores can vary independently of each other.

As said earlier, not all researchers agree with this distinction, and some theories support the hypothesis that memory is unitary over all time scales, from milliseconds to years [Brown et al., 2007]. The main reason behind this support comes from the difficulty to find a clear boundary between working and long-term memory. For instance, Tarnow [2008b] shows that recall probability vs. latency curve is a straight line from 6 to 600 seconds, with the probability of failure to recall only saturating after 600 seconds. One would expect to see some kind of discontinuity if there were two distinct stores operating in this time frame. Earlier, Nairne and Dutta [1992] has shown that the detailed patterns of recall errors look extremely similar whether recalled immediately after learning (presumably from working memory) or after 24 hours (necessarily from long-term memory).

One biological basis can be given to working memory as the prolonged firing of neurons which depletes the readily releasable pool (RRP) of neurotransmitter vesicles at pre-synaptic terminals [Tarnow, 2008a]. The pattern of depleted pre-synaptic terminals represents the long-term memory trace and the depletion itself is the working memory. Endocytosis causes working memory to decay, after the firing has slowed down. If the endocytosis is allowed to finish (the memory is not activated again), the patterns of exhausted post-synaptic terminals become invisible and the working memory disappears. The long-term memory remains as the meta-stable pattern of the neuronal excitations.

[20]Anterograde amnesia is a loss of the ability to create memories after the event that caused the amnesia occurs.

[21]A famous example is Henry Gustav Molaison , better known as patient HM or H.M. He was a memory-impaired patient who was widely studied from the late 1950s until his death. His studies played a crucial role in the development of theories that explain the link between brain function and memory, and in the development of cognitive neuropsychology [Squire, 2009].

[22]One common distractor task involves: repeatedly subtracting a single-digit number from a larger number following learning.

Other models have tried to explain the reason behind the limited duration of working memory. One hypothesis is the spontaneous decay of working memory contents over time. This hypothesis has been part of many theories of working memory (most notably the working memory model of Baddeley and Hitch [1974]). Most models working on this hypothesis also suggest that a mechanism should exist to work around this decay. Usually, it is suggested that this is achieved by rapid cover rehearsal. The idea here is that, in order to overcome the limitation of working memory (and retain information longer), information must be periodically repeated (or rehearsed). This can be done by many means, i.e. articulating it aloud, mentally simulating such articulation, etc. This allows the information to re-enter the working memory store and be retained for a bit longer, until decay or rehearsal.

But again, all researchers do not agree with this hypothesis; some say that spontaneous decay does not play any significant role in forgetting [Lewandowsky et al., 2004; Nairne, 2002], and found the evidences to be not conclusive [Jonides et al., 2008]. Following that train of thought, authors have offered an alternative form of interference: the simultaneous presence of multiple elements (i.e. digits, words, pictures, . . .) in the working memory store creates a competition between those elements. In other words, each element degrades another element present in the store. This enforces the idea that new contents gradually push out older ones. The only way to preserve a memory, in the store, and shield it against interference is by rehearsal or directing attention to it [Oberauer and Kliegl, 2006].

Before the term working memory was coined, this memory was referred as the short-term memory. Since then the term short-term memory is more used to describe memory formed in the hippocampus, since it describes them more accurately. Long-term memories are formed in the hippocampus, but before they can be stored permanently in a long-term memory store they persist in the hippocampus (during a period up to two month). It has been shown that those memories are clearly not permanent, yet persist much longer than the usual working memory. The capacity of this store does not show the limitation of the working memory either. It has been also proved that this memory store is genuine, since brain damage can cause long-term memory loss without affecting memory recently learned (which are clearly still in the hippocampus). The hippocampus and its role in memory is covered in much greater details in section 4 on page 26.

3.3. Long-term Memory. In contrast to sensory and short-term memory, long-term memory indefinitely stores a seemingly unlimited amount of information. It also differs structurally and functionally from short-term memory [Peterson and Peterson, 1959]. A simple example is phone numbers: it is only possible to remember a seven-digit number (without rehearsal) for only a few second, before forgetting. However, given a context and meaning to this number (i.e. a phone number of seven digits) it can be remembered for years, if there is some repetition; the information is said to be stored in long-term memory. Where working memory mainly stores information acoustically, long-term memory encodes it semantically. For instance, Baddeley [1966] found that, after a period of time (20 minutes), participants had less difficulty recalling words that had a similar meaning (e.g. big, large, great, huge, . . .). There is however a debate, since not all evidences support semantical encoding. The main structural difference between the long-term memory store and the other ones is that it is not unified, but spread across several regions of the brain.

As stated earlier, rehearsal and meaningful association is believed to be part of the process which allows a memory to be moved from working memory store to long-term store. The underlying biological mechanisms of long-term memory are still unclear, but the process of long-term potentiation (which involves a physical change in the structure of neurons) has been proposed as the mechanism by which working memories become long-term memories. Long-term memory is also subject to fading in the natural forgetting process, and it needs several recalls (retrievals) of a memory in order for it to last for years in long-term memory store (also dependent on the depth of processing).

Some theories consider sleep to be an important factor in learning and long-term memories. It has been considered that sleep consolidates and optimizes the layout of implicit procedural memories[23] [Robertson et al., 2004]. Other studies [Wagner et al., 2004] found that after sleep there is an increased insight, that is, a sudden gain of explicit knowledge (e.g. finding a hidden abstract rule underlying all sequences to be recalled). This implies that during sleep the representations of new memories are restructured.

Long-term memory is too generic to properly describe the structures underlying it in the brain. As stated, this store is not unified; and long-term memories are stored across several regions. Current views believe the long-term memory to be divided into two major groups: the explicit memory (also known as "declarative memory") and the implicit memory (also called nondeclarative memory).

Declarative memory refers to all memories that are consciously available. These memories are encoded by the hippocampus, entorhinal cortex, and perirhinal cortex, but consolidated and stored elsewhere in the brain. The temporal cortex has been proposed as a likely candidate for storage place of those memories but this is still unproven. Amongst declarative memories, two further distinctions exist. There is the *episodic memory*, which refers to memories of specific events in time, and the *semantic memory*, which refers to knowledge about the external world (e.g. the function of a pencil).

Nondeclarative memory is different from explicit memory in that it does not require conscious thought. This memory is behind the ability to do things by rote, and is not always easy to verbalize, since it flows effortlessly in human actions. Two further distinctions also exists in nondeclarative memory. First, there is the *procedural memory*, which refers to the use of objects or movements of the body (e.g. ride a bicycle). This type of memory is encoded and believed to be stored by the cerebellum and the striatum. Second, there is *priming*, which comes from human ability to recall more quickly information acquired recently (but outside working memory memory span) or repeatedly. For instance, when asked to name an American city that starts with the letters Ch, most people will think of Chicago, unless they have a close personal connection to or recent experience with another Ch city (Charlotte, Cheyenne, Charleston).

Craik and Lockhart [1972] proposed that it is the method and depth of processing that affects how an experience is stored in memory, rather than rehearsal. Several experiments by other authors support this idea:

- Mandler [1967] showed that *organization* play an important role in learning. In his experiments, the test subjects were given a pack of cards with one word on each, and were told to sort them any way they liked. Later during the recall process, the participants were asked to remember as many words as possible. Results pointed out there was a correlation

[23] One particular type of long-term memory, see next paragraphs for more details.

between the number of categories and the number of words the subjects could recall. Which in turn indicates that the act of organizing information facilitates its recall.

- Tyler et al. [1979] conducted an experiment where people memorized words presented to them as a series of anagrams, which they have to solve first. Experiments show participants recalled tougher anagrams (e.g. HREFAT) with more success than easier one (e.g. FAHTER). This, in turn, leads to the assumption that *effort* correlates with memorization.
- Eysenck and Eysenck [1980] tested how memory recall could be affected with the *distinctiveness* with which they were learned. To this end, they asked some subjects to spell aloud the words they had to learn, while others were simply asked to read them off a list. The first group of participants was much more successful than the second one.
- Palmere et al. [1983] gave participants descriptive paragraphs of a fictitious African nation. The paragraphs were either short or with extra sentences elaborating the main idea. They recorded a higher recall for the idea from the subject, which were given the elaborated paragraphs. This indicates that *elaboration* also helps memorization.

3.4. Working Memory Model. Models of memory provide abstract representations of how memory is believed to work. Several models have already been mentioned, this part will take a closer look at working memory models. Working memory is a theoretical framework that deals with structures and processes used for temporarily storing and manipulating information.

Many theories exist, some deal with the theoretical structure of the working memory [Miller et al., 1960], while other ones are focused on the role of specific parts of the brain involved in working memory [Curtis and D'Esposito, 2003; Kane and Engle, 2002]. There is a general agreement that the frontal cortex, parietal cortex, anterior cingulate, and parts of the basal ganglia are crucial for its functioning. The neural basis of working memory mostly comes from lesion experiments in animals and functional imaging upon humans.

There exists a relationship between short-term memory and working memory; while authors do not always agree and often describe it differently, they generally concord that the two are distinct concepts. The main difference between the two is that the short-term memory does not generally refer to the manipulation or organization of material held in memory; it is only concerned by the storage of information. Hence, the use of appellations such as: short-term memory store or short-term store. Working memory models imply the presence of a short-term memory while short-term memory is independent of this more hypothetical concept.

The term "working memory" was coined by Miller et al. [1960] in the context of theories that compared the mind to a computer. This term was also used later by Atkinson and Shiffrin [1968] to describe their "short-term store", so that the use that they make is no longer appropriate and it is best to refer to it as a short-term memory. Most researchers today have replaced the older concept of short-term memory by the concept of working memory, which includes the short-term memory. This puts a stronger emphasis on the notion of information manipulation, instead of a passive maintenance. The term short-term memory is now used to describe the hippocampal memories.

Amongst working memory models, three of them have received the distinct notice of a wide acceptance.

The first model was proposed by Baddeley and Hitch [1974], and made popular the multi-component model of working memory. Their model had two components (the "slave systems") which are responsible for the short-term maintenance of information and one component (the "central executive") which is responsible for the supervision of information integration and for coordinating the slave systems. One of the slave systems is called "the phonological loop" and stores phonological information[24]. It is called a loop, because it prevents the decay of its information by continuously articulating it and thus refreshing it in a rehearsal loop. As long as this loop is maintained, information can be retained there, indefinitely. The second slave system is called "the visuo-spatial sketchpad" and it stores visual and spatial information, as its name indicates. Manipulation and construction of visual images or representations of mental map can be done with this sketchpad. It can further be decomposed into two sub-parts, the visual subsystem deals with shape, color, texture, hile the spatial subsystem deals with location. Finally, the central executive component directs attention to the relevant information, suppressing irrelevant information and inappropriate actions. It also coordinates the cognitive process when multiple simultaneous tasks are done.

Later Baddeley [2000] extended the model with the episodic[25] buffer. This fourth component holds representations that integrate information from the slave systems (phonological, visual, and spatial information) but also possibly information not covered by the slave systems (i.e. semantic information, musical information). The component is episodic because it is assumed to bind information into a unitary episodic representation. This component has some similarity with the episodic memory introduced by Tulving but the main difference is that the buffer is a temporary store [see Tulving, 2002, for review on the episodic memory]

Another model has been suggested by Cowan [1995], and is called "the theory of Cowan"; it regards the working memory as a part of the long-term memory [Cowan, 2005, see also]. This also implies that representations in the working memory is a subset of representations in the long-term memory. The working memory is organized in two embedded levels. The first level consists in long-term memory representations that are activated (with no limit on the number of such representations). The second level, called "the focus of attention", is regarded as of limited capacity and holds up to four activated representations.

This model was later extended by Oberauer [2002] with a third component, which is a more narrow focus of attention that holds only one chunk at a time, and is embedded in the original focus of attention. Its purpose is to select a single chunk for processing. This allows manipulating and processing the chunk individually since parallel processing of all chunks is not possible.

Ericsson and Kintsch [1995] argued that most everyday common tasks, such as reading, require maintaining more than seven chunks of information in the memory. If the working memory had the same capacitive constraints as the short-term memory, it would be full after a few sentences and it would be impossible to understand complex relations expressed in a text. The hypothesis here is that humans accomplish such a feature by storing most of the information they read in the long-term memory store, linking them through retrieval structures. This allows the working memory to only retain a few key concepts, which are used as cues to retrieve everything associated to them by the retrieval structures. The authors have called this set of processes as "long-term working memory".

[24]The sound of language.

[25]Episodic event refers to autobiographical events (times, places, associated emotions, and other contextual knowledge) that can be explicitly stated.

Retrieval structures vary according to the domain of expertise; yet, as Gobet [2000] has suggested, they can be categorized in three distinctive typologies:

- Generic retrieval structure is developed deliberately and is arbitrary (e.g. the method of loci)
- Domain knowledge retrieval structures is similar to patterns and schemas and takes place exclusively during text comprehension.
- Episodic text structures is formed by every confirmed reader during text comprehension, if the text is well written and if its content is familiar [Kintsch et al., 1999]

Others have used this feature as a way to operationalize the long-term working memory [Guida and Tardieu, 2005; Guida et al., 2008].

4. Hippocampus

The hippocampus is a major component of the brain of humans and other mammals. It belongs to the limbic system and plays important roles in long-term memory and spatial navigation. It is a paired structure, with mirror-image halves in the left and right sides of the brain, very much like the closely related cerebral cortex. In humans and other primates, the hippocampus is located inside the medial temporal lobe, beneath the cortical surface (see Figure 2.2).

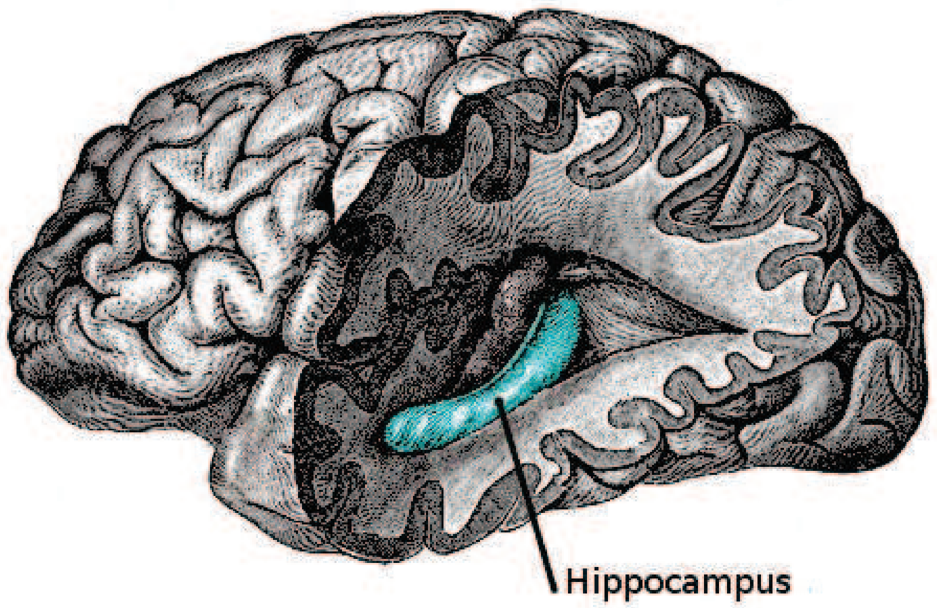


FIGURE 2.2. Lateral view of the hippocampus. The hippocampus is located in the medial temporal lobe of the brain. In this lateral view of the human brain, the frontal lobe is at left, the occipital lobe at right, and the temporal and parietal lobes have largely been removed to reveal the hippocampus underneath.

When damaged through a disease such as Alzheimer[26], hypoxia[27], encephalitis[28], or (medial temporal lobe) epilepsy[29]; the symptoms are memory problem and disorientation. With extensive damages it can also include amnesia (the inability to form or retain new memories). This corroborates the roles attributed to the hippocampus. It is essential (for learning new information) to the consolidation of information from working memory to long-term memory, although it does not seem to store long-term memory.

The hippocampus has been the main attention of extensive studies on rodents, as part of the system responsible for spatial memory and navigation. Many neurons in the rat and mouse hippocampus respond as place cells. That is, they fire bursts of action potentials when the animal passes through a specific part of its environment. Those place cells interact heavily with head direction cells (which act as internal compass) and with grid cells in the neighboring entorhinal cortex.

Given its densely packed layers of neurons, the hippocampus has frequently been used as a model system for studying neurophysiology. The long-term potentiation (LTP) was first discovered in the hippocampus and has often been studied in this structure. This neural plasticity is widely believed to be one of the main neural mechanisms by which memory is stored in the brain.

4.1. Hippocampal Function. One of the first widely accepted hypotheses about the hippocampus was its role in olfaction. It was supported by the belief that hippocampus received direct input from the olfactory bulb, which was later proven false [Finger, 2001]. Even though authors continued to investigate hippocampal olfactory response (in particular its role in olfactory memory) [Eichenbaum et al., 1991; Vanderwolf, 2001], few still believe that olfaction is its primary function.

Since then three hippocampal functions are mainly studied in the literature: inhibition, memory, and space. The inhibition theory was very popular until the 1960s, but since then it suffered criticism [Nadel et al., 1975] and has lost popularity [Best and White, 1999]. The two main evidences behind this theory are that animals with hippocampal damage tend to be hyperactive and they show difficulties learning to inhibit responses they were previously taught. Gray and McNaughton [2000], from this line of though, made a full-fledged theory of the role the hippocampus plays in anxiety .

Even if it had historical precursors, the idea relating the hippocampus to memory gained a lot of credit and took its origin in the work of Scoville and Milner [1957]. They described the results of a surgical destruction of the hippocampus[30] in a patient named Henry Gustav Molaison[31]. The unexpected result of this surgery was a severe anterograde and partial retrograde amnesia[32]. The case study of patient H.M. drove so much interest that it rapidly became the most intensively studied medical subject in history [Squire, 2009]. Following this stunning outcome, the ensuing years saw lots of studies ranging from patients with similar levels of hippocampal damage and amnesia (from accident or disease) to thousands of experiments on the physiology of activity-driven changes of synaptic connections in

[26]The hippocampus is amongst the first region of the brain attacked in Alzheimer's disease.

[27]Oxygen starvation.

[28]Acute inflammation of the brain.

[29]Epilepsy is a common chronic neurological disorder characterized by recurrent unprovoked seizures.

[30]The destruction occurred in an attempt to relieve epileptic seizures.

[31]The famous patient H.M.

[32]Retrograde amnesia is a form of amnesia where someone will be unable to recall events that occurred before the development of amnesia.

the hippocampus. There is now a general consensus that the hippocampus plays an important role in memory, even if the exact nature of this role is still widely debated [Cohen and Eichenbaum, 1993; Squire, 1992].

The third major theory relates hippocampal function to space, influenced by the work of Tolman [1948] on "cognitive maps" in humans and rats. The first work by O'Keefe and Dostrovsky [1971] led to the discovery that rat's hippocampal neurons showed activity related to its position within the environment. Later, O'Keefe and Nadel [1978] started the spatial theory of the hippocampus. Even if it was not well received at first, this theory has, since then, gained a lot of credibility. It is now regarded as one of the main theories of hippocampal function. As often, the exact details of the inner working are still debated [Moser et al., 2008].

4.2. Role in memory. There is general agreement among psychologists and neuroscientists on the importance of the hippocampus in new memory formation about experienced events[33] [Cohen and Eichenbaum, 1993; Squire and Schacter, 2002].

As explained above, damages to the hippocampus affect memory in two distinct ways. First it always causes anterograde amnesia, then it can also cause retrograde amnesia. Generally, retrograde amnesia related to hippocampal damage spares older memories, which in turn suggests that, with time, those memories are consolidated elsewhere in the brain [Broadbent et al., 2002]. However, damages to the hippocampus do not affect all types of memory, such as the ability to learn new motor or cognitive skills (e.g. playing a musical instrument or solving certain types of puzzle). This indicates that those abilities rely on other memory types (procedural memory) and are obviously located in different brain regions. An example of this arose when patients are asked to guess which of two faces they have seen most recently: even with hippocampal damages patients are able to answer correctly, even if most of the time they assert never having seen either of the faces before. From those evidences, some scientists started to distinguish between conscious recollection, which depends on the hippocampus, and familiarity, which depends on portions of the medial temporal cortex [Diana et al., 2007].

4.3. Role in spatial memory and navigation. Most successful studies come from observing rats and mice: it has been shown that many hippocampal neurons have place fields. When it comes to primates, evidence of place cells is limited, but probably only because of difficulties recording brain activity from freely moving monkeys. However, place-related hippocampal neural activity has been reported in monkeys moving around inside a room while seated in a restraint chair [Matsumura et al., 1999]. Other studies [Rolls and Xiang, 2006], focus on describing hippocampal cells that fire in relation to the place a monkey is looking at, rather than the place its body is located. Ekstrom et al. [2003] conducted an experiment on humans. They reported the presence of place cells in patients with drug-resistant epilepsy, who were undergoing an invasive procedure to localize the source of their seizures. They were implanted with diagnostic electrodes in their hippocampus and moved around a virtual reality town.

Over the past decades, hundreds of experiment have been conducted on the place responses in rats and mice [Moser et al., 2008]. The majority of neurons in the densely packed hippocampal layers are pyramidal cells, and granule cells in the dentate gyrus; both neuron groups show place cell responses. Other neurons, which are inhibitory interneurons, also show significant place-related variations in firing

[33]Episodic or autobiographical memory.

rate, but much weaker and not always. There does not seem to exist any correlation between spatial topography in the environment and in the hippocampus. Place cells are known to be mostly silent when the animal is outside the place field they cover. When the animal enters this place field, they tend to reach firing rate close to 40Hz (near the center of the place field). A rat's location can be reconstructed with high confidence with the neural activity of 30-40 randomly sampled place cells. From this it can be shown that the size of place fields varies in a gradient along the dorso-ventral axis of the hippocampus, with cells at the dorsal end showing the smallest fields, while cells at the ventral tip fields cover the entire environment [Moser et al., 2008]. Smith and Mizumori [2006] showed that in some cases the firing rate of the rat's hippocampal cells also depended on the direction of movement, the destination toward which it is traveling, or yet other task-related variables.

Another hypothesis that has been proposed since the discovery of the place cells (in the 1970s) was that the hippocampus might in fact act as a cognitive map, in other words, a neural representation of the layout of the environment [O'Keefe and Nadel, 1978]. Several evidences support this hypothesis. The first evidence comes from patients with damaged hippocampus: getting lost is one of the most common symptoms of amnesia. Those patients seem to forget where they have been and how to get where they are going [Chiu et al., 2004]. The second evidence for this hypothesis comes from studies with animals, these studies have shown that an intact hippocampus is required for some spatial memory tasks, particularly ones that require finding the way to a hidden goal [Morris et al., 1982]. The "cognitive map hypothesis" has been further advanced by recent discoveries of head direction cells, grid cells, and border cells in several parts of the rodent brain that are strongly connected to the hippocampus [Moser et al., 2008; Solstad et al., 2008].

Brain imaging, performed on subjects doing a computer-simulated "virtual" navigation task [Maguire et al., 1998], showed that the activation of the hippocampus is correlated with the efficiency of performing the navigation task. Hippocampus also seems to play an important role in finding shortcuts and new routes between familiar places. Maguire et al. [2000] studied the taxi drivers of London and concluded there was a positive correlation between the volume of their right hippocampus and the length of time they had spent as taxi driver. However, the total volume of the hippocampus remained constant. This indicates that this increase is done at the expense of the anterior portion of the hippocampus, but there is no known detrimental effects reported from this disparity in hippocampal proportions.

4.4. Anatomy. Anatomically, the hippocampus is an elaboration of the edge of the cerebral cortex [Amaral and Lavenex, 2006]; it is a zone where the cortex narrows into a single layer of very densely packed neurons, which curls into a tight "S" shape. The structures that line the edge of the cortex make up the so-called "limbic system", which includes the hippocampus, cingulate cortex, olfactory cortex, and amygdala. Even if, at first, the concept of a unified limbic system was suggested, most neuroscientists have, since then, rejected this idea [Kötter and Stephan, 1997].

The hippocampus (see Figure 2.3 on the following page) consists in two major parts, the ventral and the dorsal, both of which share a similar composition, but are parts of different neural circuits [Moser and Moser, 1998]. It is divided into three regions: the dentate gyrus, the subiculum and the cornu ammonis. This last one has four subregions called CA1 through CA4[34].

[34]This naming convention comes from its shape which has been analogized variously to a seahorse, a banana, or a ram's horn [Amaral and Lavenex, 2006]. The first one is at the origin

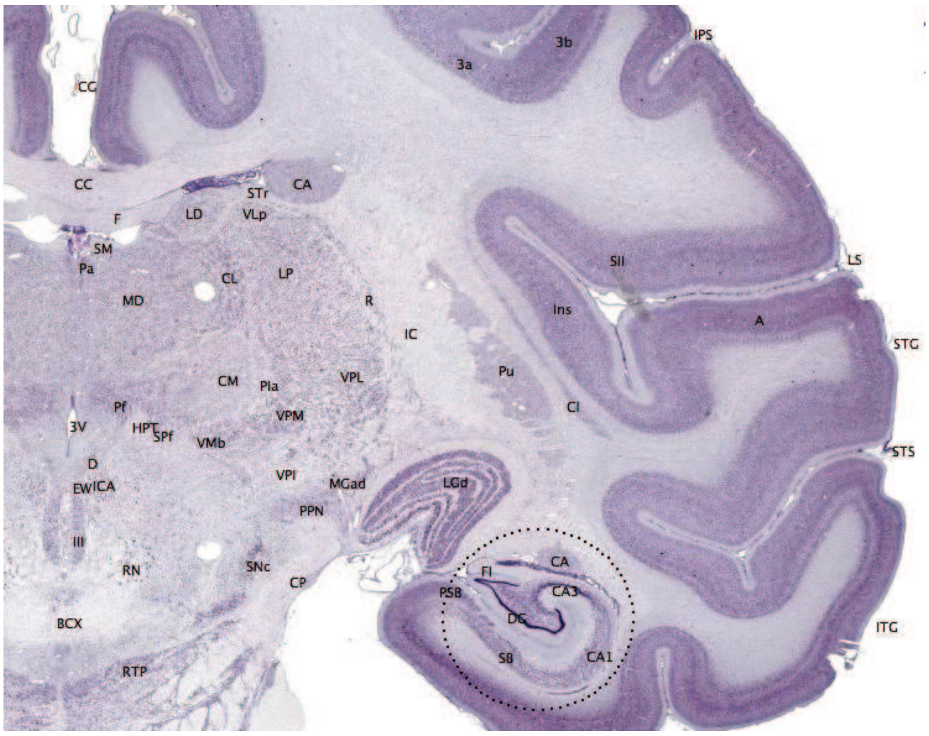


FIGURE 2.3. Coronal section of the brain of a macaque monkey. The hippocampus is shown with the circle. Source: brainmaps.org

The entorhinal cortex (EC) is the greatest source of hippocampal input and target of hippocampal output, it is strongly and reciprocally connected with many other parts of the cerebral cortex. The most prominent input to the hippocampus comes from the superficial layers of the entorhinal cortex, while the most prominent output of the hippocampus goes to the deep layers of the entorhinal cortex. Within the hippocampus, the flow of information is largely unidirectional, with signals propagating through a series of tightly packed cell layers. The signal starts on the dentate gyrus (DG), and then goes to the CA3 hippocampal subfields, then the CA1 layer, then the subiculum, then finally out of the hippocampus to the entorhinal cortex. It is important to remember that each of those layers contains complex intrinsic circuitry and extensive longitudinal connections [Amaral and Lavenex, 2006].

CA2 and CA4 are often ignored in discussions of the hippocampus. For the CA2 this is due to the fact that it is a small region located between CA3 and CA1. On the other hand, the CA4 is excluded because neurons present there do not have a pyramidal morphology like those of areas CA1 & CA3 [suggested by Lorente De N, 1934], [verified by Amaral and Lavenex, 2006; Amaral, 1978], and thus the CA4 is often grouped with the dentate gyrus[35]. The cells present in the CA4 primarily receive inputs from granule cells located nearby in the dentate gyrus

of the name, which was taken by the sixteenth century anatomist Julius Caesar Aranzi from the Greek word for seahorse (Greek: hippos = horse, kampos = sea monster). The last one is responsible for the name of the subdivision since it's an acronym for ram's horn in Latin (Cornu Ammonis).

[35]In this case it is called the hilus or hilar region

and also receive a small number of connections from pyramidal cells located in CA3. They, in turn, project back into the dentate gyrus at distant septotemporal levels.

They are not the only connections playing an important role in the hippocampal functions, other less salient connections are also present [Amaral and Lavenex, 2006; Kahana et al., 2001; Winson, 1978]

The cortical region adjacent to the hippocampus is collectively known as the parahippocampus [Eichenbaum et al., 2007], this region includes the entorhinal cortex and also the perirhinal cortex, which plays an important role in visual recognition of complex objects. However, there is substantial evidence that it makes a contribution to memory which can be distinguished from the contribution of the hippocampus, and that complete amnesia only occurs when both hippocampus and parahippocampus are damaged [Eichenbaum et al., 2007].

4.5. Physiology. The hippocampus generates some of the largest EEG signals of any brain structure because of its densely packed neural layers. It exhibits two distinct patterns either of neuronal population activity or wave of electrical activity, which can be measured by EEG. Those two major "modes" of activity are named after their EEG patterns: theta and large irregular activity (LIA). The characteristics described here are for the rat, the most extensively studied animal [Buzsáki, 2006]. Figure 2.4 on the next page shows those two modes in an EEG recording of a rat.

The theta rhythm [Buzsáki, 2002] is observed during active states , alert behavior (especially locomotion), but also during REM [36] sleep [Buzsáki et al., 1990; Vanderwolf, 1969]. During this mode of activity, the EEG is dominated by large regular waves, with frequency range of $[6, 9]$Hz. The main groups of hippocampal neurons (pyramidal cells and granule cells) show sparse population activity; this indicates that at any given short time interval, the great majority of cells are silent, while the small remaining fraction fires at relatively high rates[37]. This activity usually lasts for half a second up to a few seconds. While the rat is active, the average number of active cells is constant, even if the actual active cells change through time. As exposed above the cell activity is largely determined by the spatial location of the rat, even if other behavioral variables clearly influence this. The theta rhythm reflects sub-threshold membrane potentials, strongly modulates the spiking of hippocampal neurons and their synchronization across the hippocampus in a traveling wave pattern [Lubenov and Siapas, 2009].

The exact function of the theta rhythm is still not convincingly explained, although numerous theories have been proposed [Buzsáki, 2006]. The most popular hypothesis relates it to learning and memory [Huerta and Lisman, 1993]. Lesions of the medial septum[38] cause severe disruptions of memory, which is also an indicator. However, this must be taken with some care since the medium septum is more than just the controller of theta, it is also the main source of cholinergic projections to the hippocampus [Amaral and Lavenex, 2006].

The LIA mode is observed during slow-wave (non-dreaming) sleep, and during states of waking immobility (e.g. resting, eating) [Buzsáki et al., 1990]. During this mode, the EEG is dominated by sharp waves [Buzsáki, 1986], which are randomly timed (usually at an average rate of 1 per second), large deflections of the EEG signal lasting for 200 to 300ms. The population neural activity patterns also are

[36]Rapid Eye Movement sleep is a normal stage of sleep characterized by rapid movements of the eyes.

[37]Up to 50 spikes per second for the most actives.

[38]The central node of the theta system.

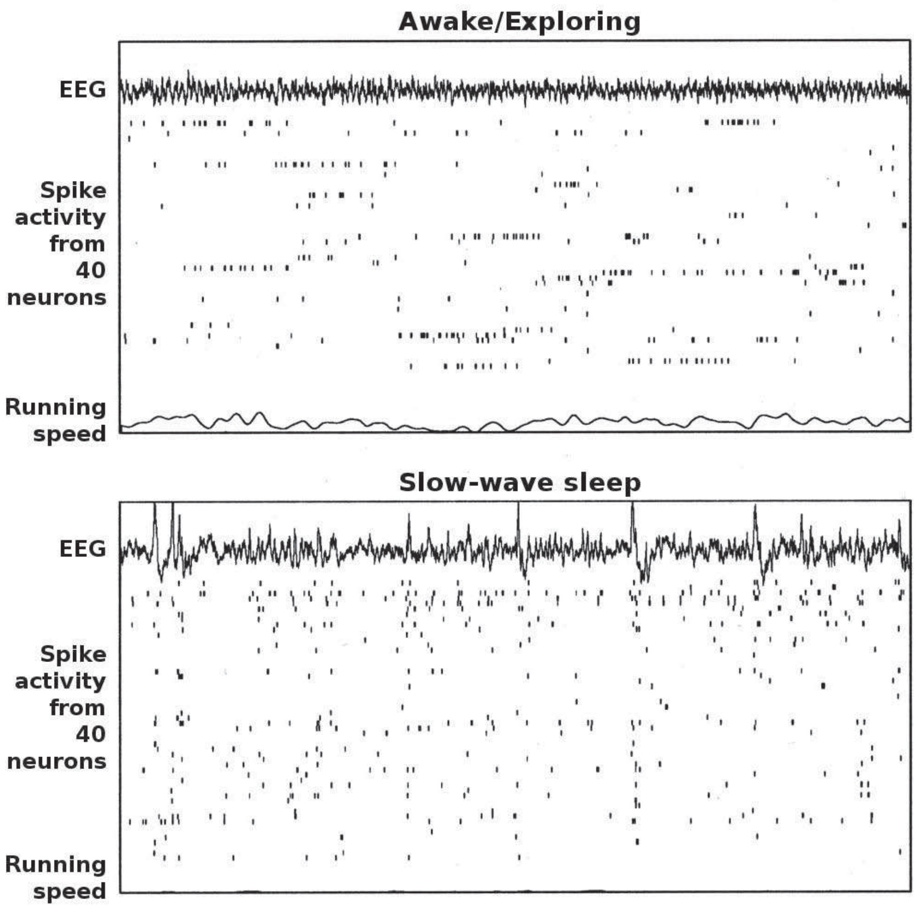


FIGURE 2.4. Hippocampal EEG and CA1 neural activity in the rats. This example shows the activity in the theta (awake/behaving) and LIA (slow-wave sleep) modes. Each plot represents 20 seconds of recording; the top part shows the hippocampal EEG trace with the continuous bold line, the central part shows the raster plot of the spikes from 40 neurons of the CA1 pyramidal cells, the bottom part shows the rat's running speed with the thin continuous line. The top plots are taken while the rat was actively searching for scattered food pellets, while the bottom ones are taken during its sleep.

determined by these waves. The pyramidal cells and granule cells are very quiet (but not silent). During this mode, 5 to 10% of the neurons, in CA3 and CA1, can emit a burst of several action potentials over a period of 50ms. These sharp waves are associated with short-lasting, high-frequency EEG oscillations called "ripples", with frequencies in the range of 150-200Hz. Amongst other things, the sharp waves appear to be associated with memory. Studies from Wilson and McNaughton [1994], like many other following studies, reported that hippocampal place cells with overlapping spatial firing fields[39] tend to show correlated activity during sleep following the behavioral session. This enhancement of correlation occurs mainly during sharp waves [Jackson et al., 2006], and is called reactivation. It has been suggested that

[39]Therefore they often fire in near-simultaneity.

sharp waves are reactivations of neural activity patterns that were memorized during behavior. This is driven by the strengthening of synaptic connections within the hippocampus [Sutherland and McNaughton, 2000]. This has been a major indication toward the idea of the "two-stage memory" theory [Buzsáki, 1989], which proposed that memories are stored within the hippocampus during behavior, and then later transferred to the neocortex during sleep. Sharp waves are suggested to drive Hebbian synaptic changes in the neocortical targets of hippocampal output pathways.

Even if all those results are observed in rats, they mostly apply to the primates. There are qualitatively similar sharp waves, and similar state-dependent changes in neural population activity, even if it does occur less frequently than in rats [Skaggs et al., 2007], but it has been difficult to see robust theta rhythmicity in the primate's hippocampus [Cantero et al., 2003], even if it is present in rabbits, cats and dogs.

5. Connectionist Models

The connectionist approach appears in several fields such as artificial intelligence, cognitive psychology, cognitive science, neuroscience and philosophy of mind. The central connectionist principle is that mental or behavioral phenomena are the emergent processes of interconnected networks of simple units. The meaning behind those units and connections can of course vary from one model to another. The most common models are neural networks were each unit represents a neuron and each connection its synapses. But other models exist; one such example might map each unit to a word and each connection to a semantic similarity between the two units (or words).

In most connectionist models, including neural network models, the network changes over time. This is often related to another very common aspect of connectionist models: activation. This implies that at any time, a unit has an activation, which is a numerical value, intended to represent some aspect of the unit. Neural network models always have some activation related in some way to the neuronal activation potential. Finally, a model is said to have a spreading activation model when a unit's activation spreads over time to other units connected to it. This is a mandatory feature for neural networks as well.

Connectionist style models have been advocated as early as in the 1940s. McCulloch and Pitts [1943] showed how one could use neural networks to implement first-order logic. They were themselves influenced by the work of Nicolas Rashevsky in the 1930s. Later, Hebb [1949] made a great contribution with the Hebbian learning algorithm, based on speculation about neuronal functionning. The theory he proposed gave the connectionism a neuropsychological base. He prones an autonomous central process intervened between sensory input and motor output. To this day, he is credited with two major contributions. First, memory is stored in connections and is learned through synaptic plasticity. Secondly, neurons do not work alone, they are organized into larger coherent configurations called "cell assemblies".

Lashley [1950] has also argued in favor of the distributed representations because of his failure to find anything like a localized engram in years of lesion experiments.

The neural network connectionist models have stressed the importance of the parallel nature of neural processing, and the distributed nature of neural representations. This has given birth to the well-known parallel-distributed processing

(PDP). This approach provided a general mathematical framework which involves eight major aspects:

- a set of processing units, represented by a set of integers,
- an activation for each unit, represented by a vector of time-dependent functions,
- an output function for each unit, represented by a vector of functions on the activations,
- a pattern of connectivity among units, represented by a matrix of real numbers indicating connection strengths,
- a propagation rule spreading the activations via the connections, represented by a function on the output of each unit,
- an activation rule for combining inputs to a unit to determine its new activation, represented by a function on the current activation and propagation,
- a learning rule for modifying connections based on experience, represented by a change in the weights, based on some variables, and
- an environment which provides the system with experience, represented by sets of activation vectors for some subset of the units.

Even though these aspects are now foundational for nearly all connectionist models, they still need to be taken with caution since they are somewhat reductionist: they assume that all cognitive processes can be explained in terms of neural firing and communication.

During the 1970s, a large amount of research led to the development of PDP; it takes its roots in the perceptron theories [Rosenblatt, 1962]. However, the results obtained with those models were not as high as expected. In addition to that, many authors had extravagant and exuberant claims over the very ambitious goals [Pollack, 1988]. Later, this field came to a brutal freeze when perceptrons were made very unpopular by the book of Minsky and Papert [1969], which elegantly demonstrated the limits of the perceptron in term of what sort of function it could calculate. It shows the perceptron cannot properly handle even the simplest functions, like the exclusive disjunction, because of its nonlinear nature.

Later in the 1980s, they gained a new recognition and popularity, with the seminal books by Rumelhart et al. [1986a,b]. They showed how to overcome the limitations exposed by Minsky and Papert [1969] with the use of multi-level perceptrons. These multi-level perceptrons are also called “nonlinear neural networks” and were reported to be far more robust and usable for a vast array of functions. They also showed that, by modifying the activation function of the units from a binary to a continuous[40] analogue threshold, it was possible to adapt the error algorithm from single layer networks to multi-layer feed forward networks by using a mechanism of back-propagation of errors to adjust the connection weights. This algorithm has been used in many practical applications for classification and data analysis.

This new wave of connectionist models was not the sole work of the PDP research group. Cybenko [1989] showed that any continuous function can be approximated using a sufficient number of hidden units. Before that, Hopfield [1982] proposed a connectionist model that worked as an associative memory, based on an

[40]Continuous neurons are required since the error back-propagation is based on the derivative of the transfer function.

analogy to a well-studied physical system: the spin glasses[41]. With this, Hopfield built a bridge with statistical physics. He was not the only one seeking inspiration from physics: Kirkpatrick et al. [1983] have ported the simulated annealing to connectionist models. They define a Boltzmann Machine[42] for which a global minimum can be found using a simple learning procedure, using only local information, to interactively adjust the weights. Kohonen [1982] developed self-organizing maps used for pattern recognition, using an unsupervised competitive learning algorithm.

Naturally, these models cannot match the brain in terms of scale; even in the most complex neural model, the units are far simpler than the real neural circuitry. Nevertheless, this is not as daunting as one might think. Connectionist models have successfully replicated certain functions exclusive to the brain (i.e. learning), with a much smaller scale. From an engineering perspective, those neural networks are just technological systems for complex information processing; therefore, they are evaluated according to the performance they can provide in dealing with complex (usually highly nonlinear) problems in areas such as association, classification or prediction [Haykin, 1998].

Even if most researchers use PDP and connectionism interchangeably[43], other theoretical works exist and are good candidates to be classified as connectionist. Early works in psychology show the first traces of connectionist theory: as early as 1869, the neurologist John Hughlings Jackson argued for multi-level, distributed systems. Freud [1895] and Spencer [1872] followed his work with the first connectionist or proto-connectionist theories, which ended as speculative theories (unsupported by evidence). By the early 20th century, Edward Thorndike was experimenting on learning and posited a connectionist type network.

Hayek [1952] proposed that spontaneous order in the brain arose out of decentralized networks of simple units. His work has only become referenced in the PDP literature recently. Sydney Lamb is known as the father of the relational network theory of language[44], which is also a form of connectionist model [see Lamb, 1998, for review]. This model was never unified with the PDP approach and has only been used by linguists; this in turn affected the number of researchers working on it. Others such as [Sun and Alexandre, 1997], proposed hybrid connectionist models mixing symbolic representations with neural network models.

Neural networks are by a large margin the most commonly used and most studied connectionist model today. There is a lot of different neural network models in the literature but they usually follow these two rules. First, any mental state can be described as a N-dimensional vector, whose value represent each neurons activation value[45]. Secondly, memory is generated by modifying the connection's strength[46] between the neurons, generally represented in a $N \times N$-dimensional matrix. This does not mean all neural networks are alike; a lot of variety can come from the interpretation of units, the definition of activation, the learning algorithm, ...

[41]Spin glasses are metal alloys with a small number of magnetic impurities arranged randomly in the alloy.

[42]The Boltzmann machine uses a stochastic method: the probability of a unit's next state is a function of the global parameter T, which defines the "temperature".

[43]However, the term "connectionism" is never used in the books of Rumelhart et al. [1986a,b].

[44]Which is also known as stratificational theory.

[45]As explained more in detail in Chapter 3 on page 63; this can range from a single real number to very complex structures.

[46]Also referred to as weights.

5.1. Structure of a neural network. A connectionist model has three defining parts. They are the signal processing elementary units (formal neurons), then the parallel connections of those processing units (which defines the architecture), and finally those connections are weighted and vary following some learning mechanisms.

The formal artificial neuron is a very simple abstraction of the real neuron (shown in Figure 2.5). It is a functional unit, whose output varies depending on several parameters. Of course, the chosen activation function[47] plays a huge role in the output of the unit, but its current state and the value of its inputs (the states of other neurons feeding this one) also affect the output. This here simulates the basic functions of a neuron, as it is fed by various inputs, each input is weighted through its connection to the unit, these weighted inputs are put together (usually through a simple sum), this gives the neuron's potential. A threshold function may be added to the end of the line; in this case, the potential feeds the (usually nonlinear) transfer function giving the output of the neuron. This describes the general process of the activation of a unit.

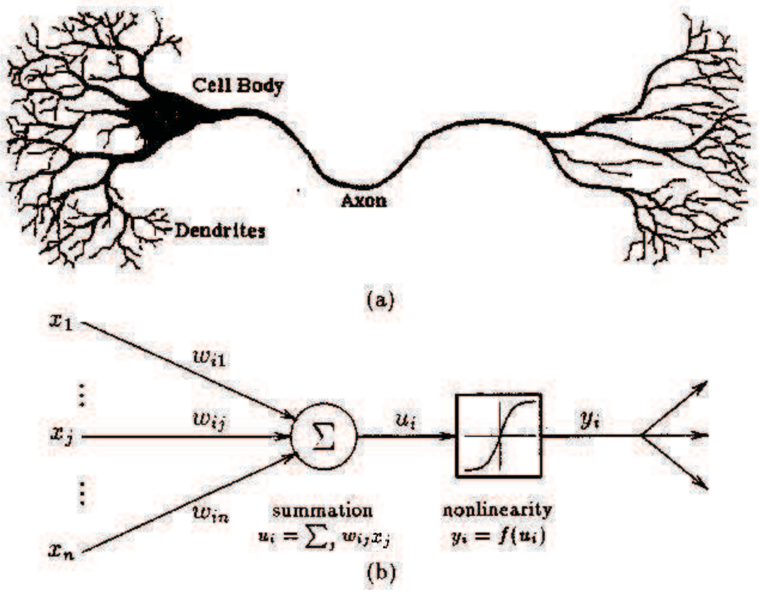


FIGURE 2.5. (top) The stylized nerve cell, (bottom) the formal unit of a neural network.

The transfer function plays a big role in the type of dynamics a neuron can describe, and therefore, the kind of information it can encode in the sequence of action potentials it emits. Many kinds of threshold functions have been used over the years. Historically, the first one was the *threshold gate*. Introduced by McCulloch and Pitts [1943], this function has an all-or-nothing behavior, which in turn makes the neuron's activation discrete. In the model presented in their paper, they used the neurons as binary devices, there was either an occurrence or absence of spikes, which in turn was used to encode the information. Despite it simplicity, the threshold gate has been used in some of the major connectionist models, such as the perceptron and multi-layer perceptron, the famous Hopfield network, the Boltzman machine, ... This transfer function was really appealing due to its computational

[47]In this figure the activation function is $f(u_i)$.

power but lacks biological plausibility; in addition to that, the dynamics that can be generated with a discrete function are much more limited.

The next generation of threshold functions was the *rate of discharge*. It is a continuous activation function, often sigmoid-like in shape. This closes the gap with biological evidences since it can be mapped to the firing rate of a neuron. These functions open the door to a world of possibilities. More powerful learning procedures can be defined using continuous activating neurons, such as the back-propagation of error [obtained by Widrow and Hoff, 1960] along with the multiple layers of a feed forward network. It has also been proven that the nonlinear transfers function could, in theory, allow a neural network to compute any analog function [Cybenko, 1989]. These two definitions of formal neuron lack any explicit use of time however. While in the first model, time is completely missing, in the second one, it appears with the biological interpretation. In both cases, this allows a discrete implementation

The final class of formal neurons that will be covered here is the spiking neurons and *the integrate-and-fire* models. The need for these models came from the observation that even though firing rate plays a crucial role in the nervous system, some biological neurons rely on the exact timing of individual spikes. A simple yet important example is the synchronization of the neurons; if they fire at the same rate it indicates that they have the same frequency; but without a precise timing of spikes, it is not possible to know if the two neurons are in phase or not. These models have been introduced to take care of these kinds of problems by explicitly integrating time in the activation process [Maass and Bishop, 1998]. In contrast with the first models, time is a crucial component in the activation of a neuron in those models.

Given a model of formal neuron, to have a connectionist model one needs to define an architecture (or topology) which will describe how the neurons are arranged and how they are connected to each others. In the literature, the number of different architectures is beyond the scope of this work, and there is no absolute and generally agreed way of categorizing them. However, from a topological point of view, there are two major groups of neural networks (see Figure 2.6):

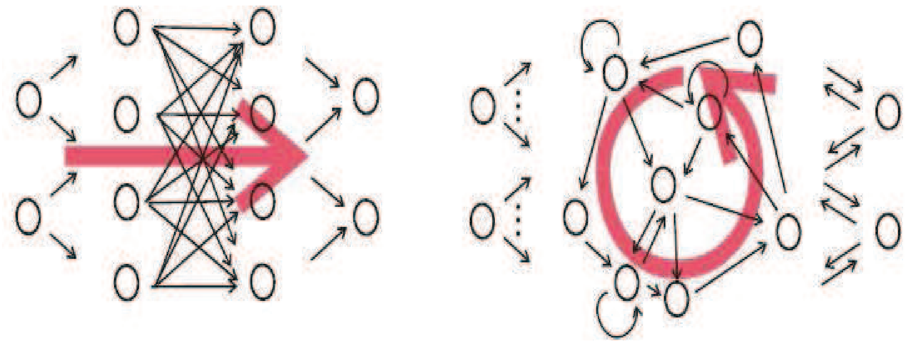


FIGURE 2.6. The two main classes of neural networks (connection wise). On the left, a feed forward neural network is modeled as a directed acyclic graph structure. On the right, a recurrent neural network is modeled as a graph structure, which includes at least one cycle (of any order).

5.1.1. *the feed forward neural networks.* These networks are represented with a directed acyclic graph. The activation is "piped" through the network from input units to output units. A biological interpretation appears in the retina, where a hierarchical feed forward cortical architecture is used for the preprocessing of visual information.

It is easily possible to define layers for those graphs since they include no cycle. The first layer, usually called input layer, is composed of all the neurons with no incoming connections. The output layer is composed of all the neurons that have no outgoing connections. The remaining neurons are divided into hidden layers. A network can have up to $N-2$ hidden layers, where N is the number of neurons in the network. All layers can be labeled with a number from 0 to K, where K is the number of hidden layers $+1$, and so the layer 0 is the input layer, the layer K is the output layer, and the layer $0 < i < K$ is the hidden layer i. From this definition, a neuron is in the hidden layer i, if and only if, it is not an input or output layer neuron and all its incoming connections are from neurons present in layers $j < i$. Figure 2.7 on the next page shows an example of layer decomposition of a feed forward network.

Depending on the learning algorithm used, feed forward networks can be divided into many groups. The most common ones include the multilayer perceptron (MLP), the radial-basis function (RBF) network, the principal component analysis (PCA) network and the self-organizing map (SOM). Feed forward networks by themselves are nonlinear static networks. They can become nonlinear dynamical systems by incorporating short-term memory in their input layer. Those networks are sometimes called "the focused time-lagged feed forward network" (TLFN) (e.g. the Tapped-Delay-Line (TDL) memory). An attractive characteristic of nonlinear dynamical systems built in this way is their inherent stability, since they are built on top feed forward network.

5.1.2. *the recurrent neural networks.* These networks include at least one cyclic path (of any order). Due to the presence of cycles, it is no longer possible to divide the network in layers. Connectionists often consider them a better model of the brain than the feed forward neural networks (even if some parts of the brain are acyclic, they often seem to be feedback processes in places). More and more models are starting to incorporate dynamical systems theory in these networks, since they can naturally be described as such. Many researchers argue that connectionist models will evolve towards fully continuous, high-dimensional, nonlinear, dynamic systems approaches.

Recurrent neural networks inherently implement short-term memory by allowing the output of a neuron to influence its input, either directly or indirectly via its effect on other neurons. This allows the network to reflect the input presented to it but also its own internal activity at any given time. All biological neural networks are recurrent, which makes those networks a good choice. The pioneer work with recurrent neural networks is attributed to Hopfield [1982], his network had no special input or output layer, each neuron of the system was an input and output neuron (which defines an auto-associative memory). The network had connections from all neurons to all other neurons (with symmetrical weights). He showed that this model defines a dissipative system for which it was possible to define an energy function, always decreasing along dynamic evolutions; this function's shape showed some local minima. Those minima are attracting fixed points with varying basins of attraction. Retrieving a memory from a partial input corresponds to starting somewhere high on the landscape, and falling into the nearest minimum.

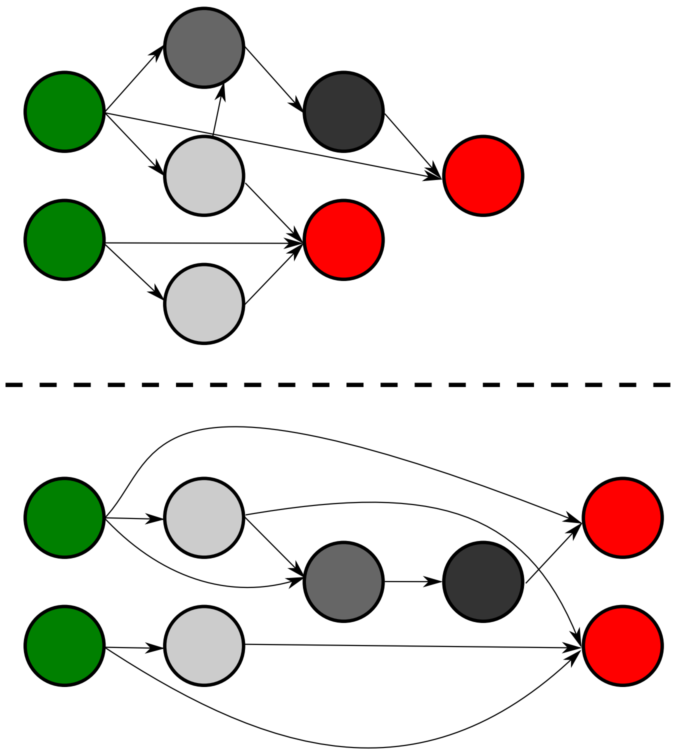


FIGURE 2.7. Decomposition of a feed forward neural network into layers. The neurons are color coded to match their layer. Input layer is green, output layer is red and hidden layer are colored as gray levels from light to dark. The top part of the figure shows the network with nodes placed arbitrarily while the bottom part shows the same network were nodes are organize to match their layer.

It is obvious that cognitive processes and/or more practical applications will require higher-level architectures. This is a solid reason to investigate recurrent neural networks even if feed forward networks showed good results in many practical applications in different areas, from classification to time-series prediction.

5.2. Learning. Learning is a crucial part of connectionist models: since the introduction of those models, many learning algorithms have emerged, ranging from extremely simplistic procedures to highly sophisticated ones. It is one of the most challenging problems in neuroscience. A learning process defines a mechanism by which the connection weights of the neural network are adapted, through a continuing process of stimulation by the environment in which the network is embedded [Haykin, 1998]. However, this modification process can take very different forms. It can be a simple one time process based solely on the data set one wishes to learn, or it can be a continuous dynamic process which alters the weights of the system in a completely unsupervised way according to various parameters of the system (i.e. feeding stimulus, current state, previous mental activity, environment,...). One very common approach was the back-propagation algorithm which used a gradient descent learning, involving changing each weight from the partial derivative of the error surface with respect to the weight [Werbos, 1974, 1994].

It is possible to classify learning algorithms regarding one important factor: **the supervision**. Those algorithms are called therefore, "supervised" and "unsupervised". Supervised learning assumes the availability of a labeled set of training data made up of N input-output examples. It requires the computation of the neural network's weights so that the output from the activation of the networks by a given input is, in a statistical sense, close enough to the desired output for all inputs in the learning data set. One example is to use the mean-square error as the index of performance to be minimized. Unsupervised algorithms do not rely on any teacher or specific input/output data set to operate. One way to make an unsupervised learning is through the adjustment of the synaptic weights.

Another way to classify learning algorithms is: **the locality**. A learning process is local if it can update the synaptic weights of a neuron with only the help of the information available to that particular neuron. The most famous local algorithm is still to this day the Hebbian rule [Hebb, 1949], which suggests to strengthen a neural connection each time units at each side of this connection are active together, or decrease it if only one is active at a given time. Locality is the key in developing parallelizable learning procedures, one of the most fundamental features of the brain. On the other hand, global rules take into account the global characteristics of the network's activity. Any update uses information from the whole system, which may be inaccessible to that particular neuron. This can help to make very smart algorithms but, unfortunately, from a biological point of view these rules are very unlikely. The back-propagation algorithm is the most common example of supervised, global learning process. Usually, it is used on multilayer perceptrons or radial basis function networks. The main idea is to propagate back the error between the current output and the desired output, from the last layer to the input layer (backward). This error propagation is used to adjust the weights of the connections. This algorithm has seen many practical uses but unfortunately it is highly unlikely since it requires a global aspect (input and hidden layers need to know the status of the output layer) but also because it is based on an instant backward propagation of information amongst the neuron, which is not possible. This algorithm has later been extended to work on recurrent neural networks by unrolling the network through time. It is then called "back-propagation through time" (BPTT) and is used in time-series prediction.

5.3. Connectionism vs. computationalism. With the rise in popularity of the connectionist model in the late 1980s, some researchers (such as Jerry Fodor or Steven Pinker) started to react. They were afraid that connectionism was dismissing what they saw as the "progress being made" in the fields of cognitive science and psychology by the classical approach of computationalism. Computationalism is a specific form of cognitivism, which argues that mental activity is computational. In other words, that the mind operates purely by performing formal operations on symbols (much like a Turing machine). The reason behind those fears was the trend behind connectionism, which was a reversion towards associationism and the abandonment of the idea of a language of thought. While this seems wrong to those researchers, others were attracted to connectionism for this very same reason.

Even if those two approaches could coexist without being at odds, the debates that started in the 1980s put the two in complete opposition. Some researchers tried to argue for the compatibility of the two approaches but never get a consensus. The main differences usually pointed out are:

- **Level of design**: While connectionism models the brain at a low level which mimics the neurological structure and evidences, computationalism

posits symbolic models that do not care for the underlying structure they model.
- **Structures and rules**: Computationalism relies on structures of explicit symbols (mental models) and syntactical rules for the internal manipulation, whereas connectionism focuses on learning from environmental stimuli and storing this information in the connections between neurons.
- **Symbolic manipulation**: The two approaches disagree on the mental activity explicit symbol manipulation; connectionists see this as a poor model of the mental activity.
- **Specialization of learning**: In connectionist models, learning is based on one (or a small set) of generic learning mechanisms, while computationalism posits domain specific symbolic sub-systems designed to support learning in specific areas of cognition (e.g. language, intentionality, number).

Despite these differences, some theorists have suggested that maybe the organic brain's implementation of a symbol manipulation system simply fits the connectionist approach. This seems plausible since connectionist models can implement symbol manipulation systems of the kind used in computationalist models, otherwise they would not be able to explain the human ability to perform symbol manipulation tasks. However, this does not dismiss the debate on whatever the symbolic manipulation forms the foundation of cognition in general.

The progress in neurophysiology and the general advances in the understanding of neural networks, has led to the successful modeling of many great problems suggested by the computationalist approach as a limitation of the connectionist models. Some of them have not been solved completely in a brain-like way and thus were considered biologically implausible. Nevertheless, today, the debate about fundamental cognition has largely been decided amongst neuroscientists in favor of connectionism. The appealing part of computational descriptions is its relative ease of interpretation, which is a weak part of the connectionist approach whose description is generally opaque and works as a black box.

6. Dynamical Systems

The dynamical systems are mathematical formalizations of any fixed "rules" that describe a point's position in its space (e.g. mathematical models that describe the swinging of a clock pendulum). It is represented at any given time by a state (a vector of real numbers), which in turn can be represented in an appropriate state space and a fixed deterministic evolution rule, that describes the next state from the current state, characterizes the dynamical system. Small changes in the state of the system correspond to small changes in the numbers. The future of the system can only be determined by iterating the system many times. The iteration process is called "solving the system" or "integrating the system". Once a system can be solved (given an initial point), it is possible to determine all its future points; this collection of points is known as a trajectory or orbit. A detailed explanation on the dynamical systems can be found in [Devaney, 1989; Ott, 1993].

The computers help to solve large groups of dynamical systems. Before that era, solutions required sophisticated mathematical techniques and could only be accomplished for a small class of dynamical systems. The numerical methods have simplified the task of determining the orbits of a dynamical system. For simple dynamical systems, knowing the trajectory is often sufficient, but most dynamical systems are too complex to be understood in terms of individual trajectories. The difficulties arise because:

- The studied systems may only be known approximately, this may be caused by a partial knowledge of the system's parameters, or terms may be missing from the equations, or even the iterations may only be done with same approximation. Of course, approximation raises the question of the validity or relevance of the numerical solutions. To address this particular problem several notions of stability have been defined (i.e. Lyapunov stability[48] or structural stability[49]). This notion of stability implies that there is a class of models (or initial conditions) for which the trajectories would be equivalent. The operation comparing orbits, in order to establish their equivalence, changes with the different notions of stability.
- The type of trajectory may be more important than one particular trajectory. All kinds of trajectories can be obtained with dynamical system, from fixed-point attractors to chaotic attractors. It is often important to be able to identify and enumerate these classes of trajectories or maintain the system within one particular class of trajectories. Mathematical studies have been done on linear systems and systems that have their state described in two dimensions; however larger systems are highly impervious to formal studies due to their complexity and high dimensionality.
- The behavior of trajectories as a function of a parameter may be what is needed for an application. As a parameter is varied, the dynamical systems may have bifurcation points where the qualitative behavior of the dynamical system changes.
- The trajectories of the system may appear erratic (like white noise). This requires using statistics over long trajectories or many different trajectories. Understanding the probabilistic aspects of dynamical systems has helped establishing the foundations of statistical mechanics and of chaos theory.

The dynamical systems theory started with the introduction of time in mathematical and physical models, originally done by Galileo (1564-1642). Until the end of the 19th century, the deterministic view of the world was dominating. Newton (1642-1727) formalized Pascal's idea of an "explicable universe". During these periods, the mathematical and physical systems were seen as reversible in time, foreseeable and reproducible. Laplace (1749-1827) expressed this idea at its best as:

> We may regard the present state of the universe as the effect of its past and the cause of its future. An intellect which at any given moment knew all of the forces that animate nature and the mutual positions of the beings that compose it, if this intellect were vast enough to submit the data to analysis, could condense into a single formula the movement of the greatest bodies of the universe and that of the lightest atom; for such

[48]In simple terms, if all solutions of the dynamical system that start out near an equilibrium point x_e stay near x_e forever, then x_e is Lyapunov stable. More strongly, if all solutions that start out near x_e converge to x_e, then x_e is asymptotically stable.

[49]Structural stability is a fundamental property of a dynamical system which means that the qualitative behavior of the trajectories is unaffected by C^1-small perturbations (where C^1 is the class of continuously differentiable functions). Unlike Lyapunov stability, which considers perturbations of initial conditions for a fixed system, structural stability deals with perturbations of the system itself.

an intellect nothing could be uncertain and the future just like the past would be present before its eyes. (Introduction to the "Essai")

In the late nineties and early twenties, uncertainty started making its way. Clausius (1822-1888) showed the irreversibility of chemical processes and defined the second law of thermodynamics: the irreversible increase in entropy. While Boltzmann (1844-1906) explained irreversibility in physics with statistic mechanics. Later, Quantum mechanics (1920-1930) introduced unpredictability as an intrinsic feature of matter. Moreover, Gödel (1930), produced fundamental results with regard to axiomatic systems, which showed that in any rich enough axiomatic mathematical system there are propositions that cannot be proven or disproven within the axioms of the system, without the introduction of inconsistency to the system. This further supports the idea that Pascal's hypothesis of an "explicable universe" is not very likely. It was introduced as the Incompleteness Theorem[50], which ended a hundred years of attempts to establish axioms that would put the whole of mathematics on an axiomatic basis.

By studying the three bodies problem[51], Henri Poincaré described what would be known as the Hallmark of Chaos - sensitive dependence on initial conditions:

> If we knew exactly the laws of nature and the situation of the universe at the initial moment, we could predict exactly the situation of that same universe at a succeeding moment. But even if it were the case that the natural laws had no longer any secret for us, we could still only know the initial situation approximately. If that enabled us to predict the succeeding situation with the same approximation, that is all we require, and we should say that the phenomenon had been predicted, that it is governed by laws. But it is not always so; *it may happen that small differences in the initial conditions produce very great ones in the final phenomena*. A small error in the former will produce an enormous error in the latter. Prediction becomes impossible, and we have the fortuitous phenomenon. (Science et Méthodes-1903)

It was in the work of Poincaré that these dynamical system themes were developed.

In his seminal paper, Lorenz [1963] described behaviors of nonlinear dynamical systems inspired by the simple modeling of the earth's atmosphere with three variables. He described the apparition of a new dynamical behavior (at some values of the state). While the activity of each individual variable looked erratic and unpredictable, their visualization in state space showed the presence of an attractor, whose shape and regularity evoked the wings of a butterfly (see Figure 2.8 on the following page). This attractor was later called a "strange attractor" by Ruelle and Takens [1971], after observing similar attractors from turbulence analyses.

Later, Mandelbrot [1975] reported that the geometrical properties of those attractors revealed a new structure that he called "fractal". By that time, the chaos theory had started to make its way as a science. One of the most interesting

[50]There are actually two incompleteness theorems of Gödel but the second one just strengthens the first incompleteness theorem, because the statement constructed in the first incompleteness theorem does not directly express the consistency of the theory. The proof of the second incompleteness theorem is obtained, essentially, by formalizing the proof of the first incompleteness theorem within the theory itself.

[51]Nine simultaneous differential equations - considered as one of the most difficult problems in mathematical physics.

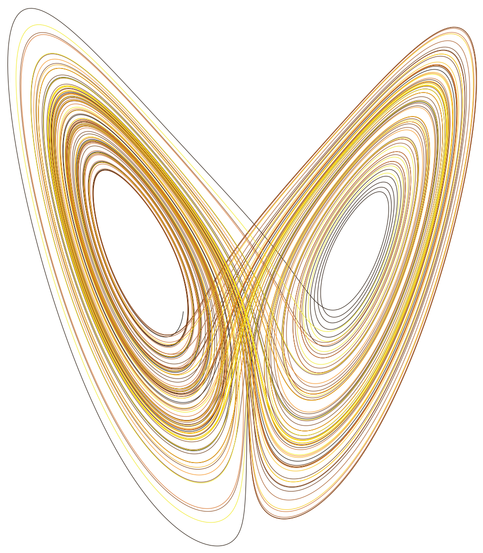


FIGURE 2.8. The Lorenz attractor, shaped like the wings of a butterfly.

findings is that order exists within chaos and determinism may lie in the apparently most disordered system.

Artificial neural networks can be looked at from a dynamical system perspective since they clearly satisfy the requirement described above. They are described, at any given time, by a state: the vector with the value of each neuron. This state can in turn can be represented in an appropriate state space: an N-dimensional space (e.g. $\{0, 1\}^N$ for the threshold gate transfer function, or $\mathbb{R}^N$ for the rate of discharge transfer function). They also have a rule which defines the next state in time: the activation function. This clearly shows that it is possible to look at a neural network as a dynamical system and this idea is not exclusive to connectionist models since neuroscientists also started to look at the brain as a dynamical system with the arrival of several new tools that support such an approach (i.e. EEG, MEG, fMRI, PET-scanners).

6.1. Definition and general properties. Dynamical systems are the study of the iteration of functions from a space into itself. This can be done in discrete repetitions or in a continuous flow of time. This choice is usually made by the studied system. When dealing with physical reality it is easier to assume a continuous space and time, which is why these systems are often described by differential equations, and thus use continuous time. On the other hand, computer made simulations are, by the nature of the support, intrinsically discrete and the use of continuous time requires faking it, as it is not available on numerical computers. This is why often discrete time systems prevail in this context. In this thesis, the choice has been made to work with discrete time models since it makes sense with computational models. Further justifications and validations of this choice are given in Chapter 3 on page 63.

There are different possible approaches to give qualitative and quantitative descriptions of dynamics. One common way is to analyze the statistical properties of the dynamical system. This field has been studied for several years [see Cessac, 2002, for review]. Another perspective is to look at the problem from a topological point of view. Here, dynamics are mainly characterized by their trajectory, hence their attractors.

6.1.1. *Dynamical system.* A discrete time dynamical system can be described as a mapping:

$$F_\rho \colon \mathbb{R}^N \to \mathbb{R}^N : \mathbf{x}(t) \mapsto F_\rho(\mathbf{x}(t)) = \mathbf{x}(t+1) \tag{1}$$

This map F describes how the system evolves through time. It is a function of a family of parameters: $\rho = (\rho_1, \rho_2, \dots, \rho_{p_N})$. By iterating the map from an initial condition x_0, the system follows a trajectory $\phi(x_0)$ in the space state $\mathbb{R}^N$. This can further be generalized by using a function $F_\rho \colon X \to X$, where X is a topological space (often a metric space).

6.1.2. *Non-chaotic attractors.* Given this definition, it is possible to define an attractor, but to this end it is first important to define the notion of neighborhood[52].

Definition 1. *In a topological space X, a neighborhood of x is a set $\mathcal{N}$ with an open sub set $\mathcal{O}$ which contains x ($x \in \mathcal{O} \subseteq \mathcal{N}$).*

Definition 2. *In a topological space X, a neighborhood of a set $\mathcal{A}$ is a set $\mathcal{N}$ which is a neighborhood of all points of $\mathcal{A}$. In other words, this means that there exists an open set $\mathcal{O}$ such that : $\mathcal{A} \subseteq \mathcal{O} \subseteq \mathcal{N}$.*

Definition 3. *$F \colon X \to X$ is said to be topologically transitive if for every pair of non-empty open sets $U, J \subset X$, there exists an integer $k > 0$, such that $F^k(U) \cap J \neq \emptyset$*

Definition 4. *For a given set U, x is a point of closure of U if every neighborhood of x contains a point of U[53]. The set of all points of closure of U is noted $\overline{U}$ and is called the closure of U*

Definition 5. *A subset U of V is dense in V if $\overline{U} = V$*

Then a non-chaotic attractor can be defined [Eckmann and Ruelle, 1985] as a non-empty closed set $\mathcal{A}$ of points in the space $\mathbb{R}^N$, with two properties:

- $\mathcal{A}$ must be attracting for a given open neighborhood $\mathcal{U}$[54] of it: $\cap_{t \geq 0} F^t_\rho(\mathcal{U}) = \mathcal{A}$
- the F_ρ-periodic orbits must be dense in $\mathcal{A}$ and $\mathcal{A}$ must contain a dense F_ρ-periodic orbit (topological transitivity).

A basin of attraction $\mathcal{B} \supseteq \mathcal{A}$ can be associated with each attractor. It represents the set of points in the state space[55], which converge towards the attractor. Given an attractor $\mathcal{A}$, the basin of attraction of $\mathcal{A}$ is defined as :

$$\mathcal{B} = \cup_{t<0} F^t_\rho(\mathcal{U}) \tag{2}$$

The different possible non-chaotic attractors are:

A fixed point attractor: is defined as a singleton attractor, $\mathcal{A} = \{x_*\}$, with the following property, $F_\rho(x_*) = x_*$. The dynamics of the system is static in this region.

[52]The neighborhood is the generalization of the open ball in metric spaces to topological space.

[53]This point may be x itself

[54]One can then choose $\mathcal{U}$ such that $F_\rho(\bar{\mathcal{U}}) \subset U$.

[55]It is also possible to extend the basin notion to neighborhoods of the parametric space.

A periodic limit cycle attractor: is defined as an attractor of $n > 1$ points $\mathcal{A} = \{x_1, x_2, \ldots, x_n\}$ such that $\forall i \in [1, n] : F_\rho^n (x_i) = x_i$ and n is the smallest positive integer satisfying this equality (this is also called the prime period, or least period, of the point x_i). When the property hold then the attractor is called periodic with period n. When the system falls in a cyclic attractor, it oscillates periodically amongst the points of that attractor.

A tore or quasi-periodic limit cycle attractor: is defined as an infinite attractor, which is given by a tore resulting from the combination of periodical functions of periods linearly independent. The system is said to iterate along a quasi-periodic limit cycle.

If the system possesses only one attractor, its basin of attraction is usually the entire state space X. However, the basins of attraction are invariant sets, in the sense that, it is clear by definition that the orbit resulting from an initial condition inside a basin of attraction is entirely in the basin of attraction. This means that when different attractors co-exist, there is a border between the two basins of attraction, called "the separatrix"[56].

6.1.3. *Deterministic chaos.* The word chaos is often misused and wrongly associated with disorder. Here, chaos can be observed in deterministic dynamical systems, and has a structure that can be characterized[57]. This structure is very interesting and possesses many intriguing properties. One such feature is the auto-similarity: this property says that zooming on a part of the chaotic domain, the same structure can be indefinitely obtained; it is also called "the fractal geometry". The trajectories on these structures are called "strange attractors". Feigenbaum [1983] says that the following characteristics are required in a dynamical system in order to observe deterministic chaos:

- **nonlinearity**: only nonlinear phenomena exhibit chaotic behavior;
- **recursivity**: the complex behavior is mainly generated by the recursive nature of the system.

Chaos has seen many different definitions over the time. Mathematicians usually use a topological approach [Devaney, 1989]. Intuitively a dynamical system is said to be chaotic in a state space when :

(1) The neighborhoods of points don't stick together in one localized clump.
(2) For any point in the state space there is a periodic orbit in its neighborhood.
(3) It is sensitive to initial conditions: an arbitrarily small perturbation of the current trajectory may lead to significantly different future behavior.

It is easy to see that the topological transitivity of F formalizes the first requirement of a chaotic system, while the density concept can be used to formalize the second requirement. To understand this, first it must be noted that a point of closure of a set U has a neighborhood that contains this set. From there it is easy to see that if the periodic orbits of the system are dense in the state space, then each point of the state space has periodic orbit in its neighborhood. The last point needs no other definition, since it can be proven that it is implied by the first two points.

[56]For example, in a simple 1-dimensional system, this border can be delimited with a simple curve (the border is the intersection of the x-axis with the curve).

[57]Even though it is possible to prove an equivalence between some deterministic chaos and a pure random process.

Theorem 6.1. [58] *If F is topologically transitive in X and its periodic orbits are dense in $V \subset X$, then F has sensitive dependence on initial conditions in V.*

Given those definitions, a formal definition of a chaotic system can be formulated.

Definition 6. *Let V be a subset of the state space X. $F\colon X \to X$ is said to be chaotic on V if:*

(1) *F is topologically transitive.*
(2) *periodic orbits are dense in V.*
(3) *F has sensitive dependence on initial conditions in V.*

Under this definition lies the three important components a chaotic system needs: unpredictability, non-decomposability and an element of regularity. The first one is given by the sensitive dependence on initial conditions. The topological transitivity prohibits the system from being broken into two subsystems[59] which do not interact under F. Finally, the density of the periodic orbits gives the element of regularity. The problem lies in the difficulty to apply concepts such as transitivity and denseness to physical systems. Generally, it is said that a physical system is chaotic if it show sensitivity to initial conditions (SIC). This property alone makes the system's behavior extremely hard to predict, since an initial small error can lead to a large divergence.

6.1.4. *The Hausdorff dimension.* It is often useful to know the dimension of an object (such as an attractor); naively one could define this as the minimum number of coordinates needed to specify each point within the space. E.g. the line has a dimension of one because only one coordinate is needed to specify a point on it. Unfortunately, this naive definition falls short when it came to analyze the dimensions of more complex objects. For example, for the famous Sierpinski triangle (see Figure 2.9), the naive approach will say that it is a 2-dimensional structure. However, a plane in the euclidean space also is a 2-dimension structure. This here clearly shows that the naive approach is too raw and the dimension of an object needs to be more accurate.

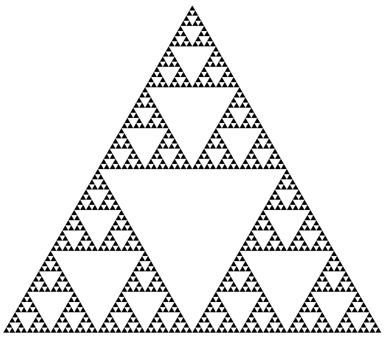


Figure 2.9. The Sierpinski triangle, fractal named after the Polish mathematician Wacław Sierpiński who described it in 1915

In mathematics, the Hausdorff dimension[60] is a non-negative real number associated to any metric space. It generalizes the notion of the dimension of a real

[58]See [Elaydi, 1999] for a proof (p.117)
[59]Two invariant open subsets.
[60]Also known as the Hausdorff-Besicovitch dimension.

vector space. The Hausdorff dimension respects the natural definition of dimension by respecting the dimension of a single point (as 0), of a line (as 1), of the plane (as 2), etc. There are however many irregular sets that have a non-integer Hausdorff dimension. The mathematician Felix Hausdorff introduced the concept in 1918, and Mandelbrot, who used the fractal dimension, made it popular.

Let $\mathcal{A}$ be a given attractor in a compact metric space. Then $N(r, \mathcal{A})$ is the minimum number of open balls of radius r needed to cover $\mathcal{A}$. The dimension of the attractor is then defined by:

$$\dim \mathcal{A} = \lim_{r \to 0} \sup \frac{\log N(r, \mathcal{A})}{\log(1/r)} \tag{3}$$

6.1.5. *Strange attractors.* A strange attractor[61] $\mathcal{A}$ is an attractor with two additional properties. First, the distance between two nearby trajectories at time t in the attractor increases exponentially. Secondly, the dimension of the attractor is fractal[62]. This clearly defines a dynamical system which evolves in an "unpredictable way" in a well-defined region of its state space. As a corollary, a chaotic attractor has $\|\mathcal{A}\| = \infty$, since otherwise the attractor will be cyclic.

6.2. Tools for dynamical analysis. The chaos theory is still very young, and it has not yet convinced all mathematicians. As Holmes [1995] said:

> I do not believe that chaos theory exists, at least not in the manner of quantum theory, or the theory of self-adjoint linear operators. Rather we have a loose collection of tools and techniques.

This section presents the most common tools in this theory. There is a very large set of tools and here are reported the ones that are used in the rest of the thesis [see Eckmann and Ruelle, 1985, for more]. Again, even though there are tools to deal with discrete valued systems, here all the presented tools work with continuous valued systems.

6.2.1. *State space.* The easiest but also the least effective way to study a dynamical behavior is through the analysis of the state space diagram. Being a graphical representation of a system's evolution towards its attractors, each point of that space unequivocally determines the state of the system at a given time. These plots usually represent time and the position on the axis. The biggest problem with this tool is that it does not scale at all with the parameter space and thus always provides a limited view of the system. In addition to that, information is hard to process; it can easily help distinguish a periodic output from a fix point but it is impossible to know what kind of complex dynamics the system outputs just with this plot. Finally, it only gives a qualitative view of the system.

6.2.2. *Return map.* A return map is the value of one variable as a function of its previous value in time, $x(t+1) = f_\rho(x(t))$. The return map[63] is a very suggestive but hardly conclusive tool for the evidence of chaos. It is easier to spot the periodicity of a variable's output on this map as it is easier to see if the output is close to white noise (or very strong chaos) or a weak chaos. Overall, this plot gives some information, is useful but not enough by itself for pretty much the same reason as the state space plot. It can only show one variable of a highly multi-variable system and it does not give any quantitative value over the observation.

[61] Which is also called a "chaotic attractor".

[62] Non integer.

[63] Also known as the Poincaré map.

6.2.3. *Bifurcation Diagram.* A bifurcation diagram shows the possible long-term values (equilibria/fixed points or periodic orbits) of a system as a function of a bifurcation parameter (also called control parameter) in the system. The bifurcation parameter describes some part of the system. For example, in the famous logistic map, defined as $x_{n+1} = r * x_n * (1 - x_n)$, r is the bifurcation parameter (and the only parameter of the system).

For a 1-dimensional system with one parameter (i.e: the logistic map): the x-axis represents the variation of the bifurcation parameter, and the y-axis represents the output of the system. First a starting condition is set (e.g. $x_0 = x_*$ for the logistic map), then for each value β of the bifurcation parameter the trajectory of the system is evaluated for N time steps (i.e: for the logistic map, the trajectory is x_1, x_2, . . . ,x_N). Then the pairs (β, x_1),(β, x_2), . . . , (β, x_N) are plotted. However, it may be useful to first iterate the system for a certain period in order to go outside of the transient period and be sure to be in the actual attractor. Figure 2.10 shows a bifurcation diagram for the logistic maps.

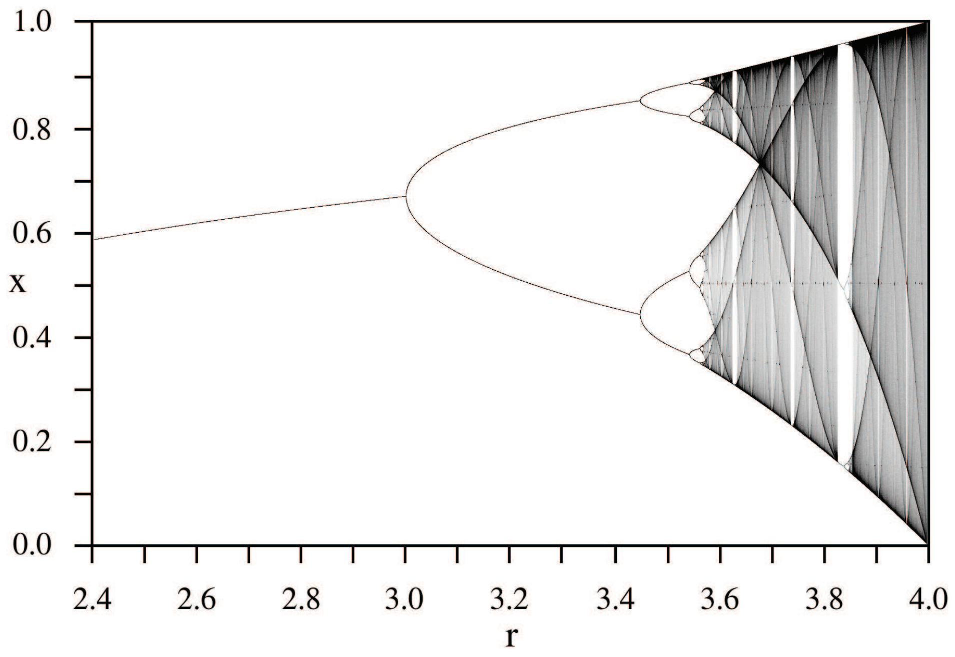


FIGURE 2.10. The bifurcation diagram of the logistic maps. The initial condition $x_* = 0.25$, $r \in [2.4, 4.0]$. For each value of r in this range (with an increment of 0.001) the map is iterated 1000 times to skip the transient phase and then 100,000 more iterations are done and plotted this time.

When the system has more than one control parameter, the bifurcation diagram can be done using any combination of those parameters. For each parameter the range of variation is set. In this case the x-axis shows the linear evolution of all the parameters (simultaneously) from their lower range to their upper range. When the system has more than one output (i.e: higher than 1-dimensional system) the observed output has to be specified, it can be any of the system's variable (or the average of all the variables).

6.2.4. *Lyapunov Exponent.* These exponents are a tool, which is extremely convenient when it comes down to determine the sensitivity of a system to small orbit perturbations (which is the easiest way to characterize chaotic attractors). To compute these exponents, the system is analyzed over two trajectories that start at very close initial conditions (which only differ by a very small offset). The Lyapunov exponent λ of a one dimensional map gives the average exponential rate of divergence. Given a small initial perturbation $\eta_s(0)$, the evolution of this perturbation after n iterations is usually given by:

$$\eta_s(n) \sim \eta_s(0)\, e^{\lambda n} \tag{4}$$

where λ is the Lyapunov exponent. From equation 4, describing the evolution of the perturbation in terms of the Lyapunov exponent, and equation 16 on page 53 describing it in terms of the eigenvalues of the Jacobian of the system, the following can be deduced:

$$\eta_s(n) \sim \eta_s(0)\, \Lambda_s^n \sim \eta_s(0)\, e^{\lambda n} \tag{5}$$

which in turn can be simplified to obtain the relation between the Lyapunov exponent and the largest eigenvalue of the Jacobian of the system:

$$\lambda = \ln \|\Lambda_s\| \tag{6}$$

This confirms that non-hyperbolic points (which are responsible for bifurcations) have a zero value Lyapunov exponent. This also shows that a m-dimensional dynamical system has m Lyapunov exponents λ_i, each of them measuring the divergence rate following one of the directions of the eigenvectors. The evolution of a hyper-volume V_0 is hence given by:

$$V = V_0 e^{(\lambda_1+\lambda_2+\ldots+\lambda_m)t} \tag{7}$$

The easiest way to compute a quick and accurate approximation of the largest Lyapunov exponent has been described by Wolf et al. [1984]. To use this on a neural network, first, the network is simulated for a sufficient amount of time steps to ensure all transient effects are avoided; this helps the system get closer to any activator that was close to the initial position. The computation of the Lyapunov exponent is done through the simulation of two close initial states of the network. Let's call them $\mathbf{x}(0)$ (where $\mathbf{x}(t)$ is a vector of all the states of the network at time t) and $\mathbf{x}'(0)$, which is the state of the network $\mathbf{x}(0)$ plus a perturbation η. Next, let's define $\Delta\mathbf{x}(t)$ as the difference between the two previous vectors:

$$\Delta\mathbf{x}(t) = \left(x_1(t) - x_1'(t), \ldots, x_n(t) - x_n'(t)\right) \tag{8}$$

which, by construction, gives $\|\Delta\mathbf{x}(0)\| = \eta$. Each of the initial states are simulated and produce $\mathbf{x}(1)$ and $\mathbf{x}'(1)$ and the variation between the initial states and the new states is given by:

$$\frac{\|\Delta\mathbf{x}(1)\|}{\|\Delta\mathbf{x}(0)\|} \tag{9}$$

The vector $\Delta\mathbf{x}(1)$ is rescaled[64] as:

$$\Delta\mathbf{y}(1) = \frac{\eta\Delta\mathbf{x}(1)}{\|\Delta\mathbf{x}(1)\|} \tag{10}$$

which means that $\|\Delta\mathbf{y}(1)\| = \eta$ and from this new value the state $\mathbf{x}'(1)$ can be recomputed as: $\mathbf{x}'(1) = \mathbf{x}(1) + \Delta\mathbf{y}(1)$. This process is repeated T times, and the

[64]This rescaling is a computational trick which has proven to be more robust on numerical computations [Wolf et al., 1984].

largest Lyapunov exponent is estimated from the average of the logarithm of the scalings:

$$\lambda = \frac{1}{T}\sum_{i=0}^{T-1} \ln \frac{\|\Delta \mathbf{x}\,(i+1)\|}{\|\Delta \mathbf{x}\,(i)\|} \tag{11}$$

Here is the interpretation from the value obtained for the Lyapunov exponent with this method:

fixed-point, $\lambda < 0$: indicates a stable attractor. If a slight modification of the state occurs, the system will converge again and the modification will vanish.

quasi-periodic points, $\lambda = 0$: indicates a limit cycle; after a perturbation, the system will continue on a nearby road.

chaotic points, $\lambda > 0$: indicates that the system is highly sensitive to initial conditions. Any perturbation will cause the network to diverge.

6.2.5. *The Lyapunov Spectrum.* In a dynamical system, the rate of separation can be different for different orientations of the initial separation vector. Thus, there is a whole spectrum of Lyapunov exponents – the number of them is equal to the number of dimensions of the state space. For a dynamical system with evolution equation f^t in a n-dimensional state space, the spectrum of Lyapunov exponents: $\{\lambda_1, \lambda_2, \cdots, \lambda_n\}$, in general, depends on the starting point x_0. The Lyapunov exponents describe the behavior of vectors in the tangent space of the state space. They are defined from the Jacobian matrix:

$$J^t(x_0) = \left.\frac{df^t(x)}{dx}\right|_{x_0} \tag{12}$$

The Jacobian matrix J^t describes how a small change at the point x_0 propagates to the final point $f^t\,(x_0)$.

The Oseledets theorem [Oseledets, 1968] shows that the limit $\lim_{t\to\infty}(J^t\cdot(J^t)^T)^{1/2t}$ defines a matrix[65] $L\,(x_0)$. If $\Lambda_i\,(x_0)$ are the eigenvalues of $L\,(x_0)$, then the Lyapunov exponents λ_i are also given by:

$$\lambda_i(x_0) = \log \Lambda_i(x_0) \tag{13}$$

If the system is allowed to run for a long time, it can be studied with the ergodic theory, which focuses on invariant measures of dynamical systems. The ergodic theorems basically assert that a system that evolves for a long time "forgets" its initial state. This also means that the set of Lyapunov exponents will be the same for almost[66] all starting points of an ergodic component[67] of the dynamical system.

Its biggest Lyapunov exponent roughly characterizes the dynamics of the system. As an example, the presence of a positive Lyapunov exponent characterizes chaotic dynamics. However, the other Lyapunov exponents are also useful and provide additional information (especially useful for differentiating chaos). For example, hyper-chaos is characterized by the presence of more than one positive Lyapunov exponent [Rössler, 1983].

[65] Here, $(J^t)^T$ is the transpose of the matrix (J^t).

[66] A property is said to hold almost everywhere, if it holds everywhere except in a set of measure zero (called a *null set*), which can be enclosed in intervals whose total length is arbitrarily small. Unfortunately some references in the literature also refer to the empty set as the null set, which is subject to confusion.

[67] An ergodic component is a component of the system that satisfies the ergodic theory.

The computation of the Lyapunov spectrum is performed using Gram-Schmidt re-orthogonalization of the evolved system's Jacobian matrix (which is estimated at each time step from the system's equations). This method is detailed in [Wolf et al., 1984].

6.2.6. *Spectral Density.* When it comes to analyze a dynamical system periodicity, the spectral density[68] is a very well known tool for the job. The power spectrum is defined as the square of its Fourier amplitude[69] per unit of time [Eckmann and Ruelle, 1985]. It measures the amount of energy per time unit contained in the signal as a function of the frequency. It is a positive real function of a frequency variable associated with a stationary stochastic process, or a deterministic function of time, which has a dimension of power per Hz, or energy per Hz. Intuitively, the spectral density captures the frequency contents of a stochastic process and helps identifing periodicities.

This tool allows to distinguish periodic, quasi-periodic and chaotic series and even distinguishes differences between chaos:

- **fixed point attractor**: yields a continuously zero spectrum;
- **cyclic attractor**: yields a peak at the frequency of the signal plus one smaller peak for each of its harmonics;
- **quasi-periodic signal**: yields peaks at the important frequencies of the periods, plus one smaller peak for each harmonic;
- **chaotic signal**: yields something in between quasi-periodic and white noise spectrum[70].

Again, this is a qualitative result with all of its drawbacks and calls for cautiousness. In addition to that, since the power spectrum analyses the signal only on a given frequency band, choosing the wrong size for this band can lead to take a band smaller than the frequency of the signal, which in turn will look like a quasi-periodic or chaotic signal. Another look at the figures 2.13 on page 56, 2.14 on page 57, 2.15 on page 58, in the light of spectrum analysis, clearly shows that the three chaos are different and distinguishable through their spectral density.

6.3. Bifurcation theory.

6.3.1. *Stability of fixed point attractors.* To determine the stability of a given fixed-point attractor x_*, it is useful to look how nearby points in space behave. This is done by adding a small perturbation to x_*, say η, and then analyzing the evolution of this perturbation:

$$F\left(x_*+\eta\right)=F\left(x_*\right)+\mathcal{D}F\left(x_*\right)\eta+o\left(\eta^2\right) \tag{14}$$

where $\mathcal{D}F$ is the Jacobian matrix of partial derivatives of F (assuming F is of class C^1). The linearized stability problem is obtained by neglecting the terms of order η^2. This gives:

$$F\left(x_*+\eta\right)-F\left(x_*\right)=\eta\left(n+1\right)\approx\mathcal{D}F\left(x_*\right)\eta\left(n\right) \tag{15}$$

where $\eta\left(n\right)$ is the perturbation after n iterations of the system (see section 6.2.4 on page 50). This reduces the stability analysis of x_* to the study of its Jacobian $\mathcal{D}F$, and therefore to the study of the eigenvalue of this matrix. The eigenvalues

[68]Which is also known as the power spectral density (PSD), energy spectral density (ESD), or power spectrum.

[69]The amplitude obtained with a Fourier transform on a signal.

[70]White noise is shown as a broad band noise.

λ_s gives the evolution of the perturbation in the direction of the s^{th} eigenvector:

$$\eta_s(n+1) = \Lambda_s \eta_s(n) \tag{16}$$

where η_s is the value of η along the direction of the eigenvector s. These eigenvalues are in general complex numbers, their norm giving the dilatation. When $|\lambda_s| < 1$ the perturbation vanishes, while $|\lambda_s| > 1$ indicates that the perturbation increases exponentially over time. The angle of the eigenvalue with the real axis specifies the speed of the rotation.

This allows to properly define the notion of hyperbolicity. A fixed point x_* for F is said to be hyperbolic if $\mathcal{D}F(x_*)$ does not contain eigenvalues of norm equal to 1. This is extended to periodic attractors: if x_* is periodic of period k, then it is hyperbolic if $\mathcal{D}F^k(x_*)$ does not contain eigenvalues on the unit circle. Furthermore, depending on the eigenvalues of $\mathcal{D}F(x_*)$, three kinds of hyperbolic points can be distinguished:

- x_* is an attractive periodic point if all the eigenvalues respect $|\lambda| < 1$.
- x_* is a repulsive periodic point if all the eigenvalues respect $|\lambda| > 1$.
- x_* is a saddle node in other cases, which has both attractive and repulsive behaviors.

Any point is non-hyperbolic as soon as it has one eigenvalue on the unit circle. These points are at the origin of the bifurcation phenomenon, which is characterized by a qualitative change in the dynamical structure of the system after a modification of a control parameter. The position of the eigenvalue on the unit circle determines one of the three possible ways a bifurcation can occur:

- $\lambda = -1$: **flip bifurcation**, corresponding to a period-2 oscillation;
- $\lambda = 1$: **saddle-node bifurcation**, corresponding to two branches of a stable equilibrium;
- $\lambda = \|a \pm bi\| = 1, b \neq 0$: **Hopf bifurcation**. During this periodic or quasi-periodic trajectory the angle of rotation α is given by the angle of the eigenvalue with the real axis. The trajectory describes a limit cycle. If the angle is a rational fraction of π, then it crosses a finite amount of points before to cycle repeatedly, which is a periodic regime. If the angle is not a rational fraction of π the trajectory never repeats and densely covers the limit cycle. The regime is said to be quasi-periodic. This kind of bifurcation is the most common in complex systems.

Figure 2.11, 2.12 on the following page show these bifurcations. These diagrams are called "bifurcation diagrams" and are obtained by modifying the control parameters of the system (on the x-axis) and plotting for each parameter change the different values by which the system goes through (on the y-axis). Those examples also show the Lyapunov exponent (described in section 6.2.4 on page 50). This exponent is very important, and amongst other things it allows to detect the occurrence of a bifurcation when it reaches 0. Incidentally this is the only way to identify a saddle-node bifurcation, since it does not appear clearly on the diagram. This is due to the fact that, from one initial condition, the system stabilizes in one of the two attractors, the path-bifurcation diagram allows to visualize this as shown on section 6.5 on page 60.

6.3.2. *Road to chaos.* A system starting on a steady state, which goes through a path of bifurcation, through a variation of control parameters (see section 6.2.3 on page 49), until it reaches a chaotic state, is said to describe a road to chaos. The change in the control parameter appears as a change in the eigenvalues of the Jacobian matrix and a bifurcation occurs when one of these eigenvalues crosses the

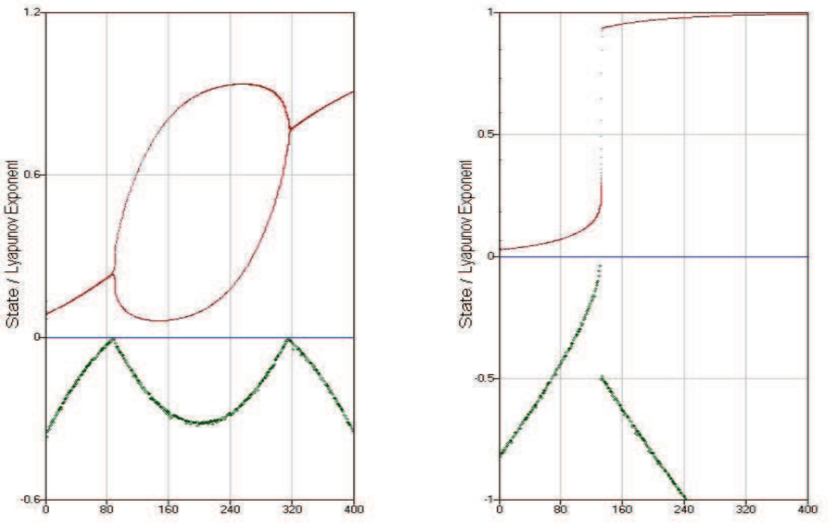


FIGURE 2.11. (left) A flip bifurcation, produced with a self-inhibiting ($\omega < 0$) one-neuron system, with a positive input ($\theta > 0$) that acts as the control parameter. (right) A saddle node bifurcation in a self-excitatory ($\omega > 0$) one-neuron network, with a negative input ($\theta < 0$) that acts as the control parameter. The Lyapunov exponent, which is equal to 0, indicates the bifurcation. The corresponding Lyapunov exponents are plotted in green. The evolution function is $x\left(t+1\right) = \theta + \omega\left(\tanh\left(x\left(t\right)\right) + 1\right)/2$.

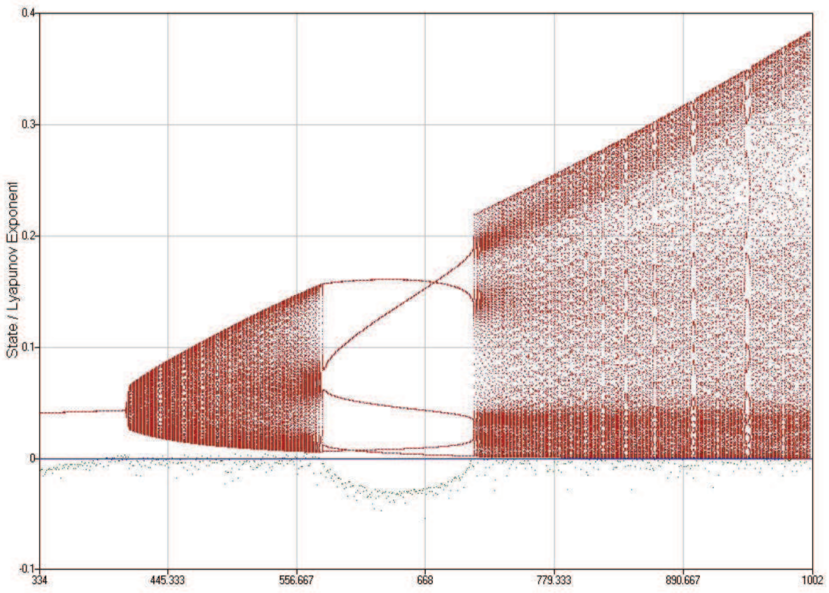


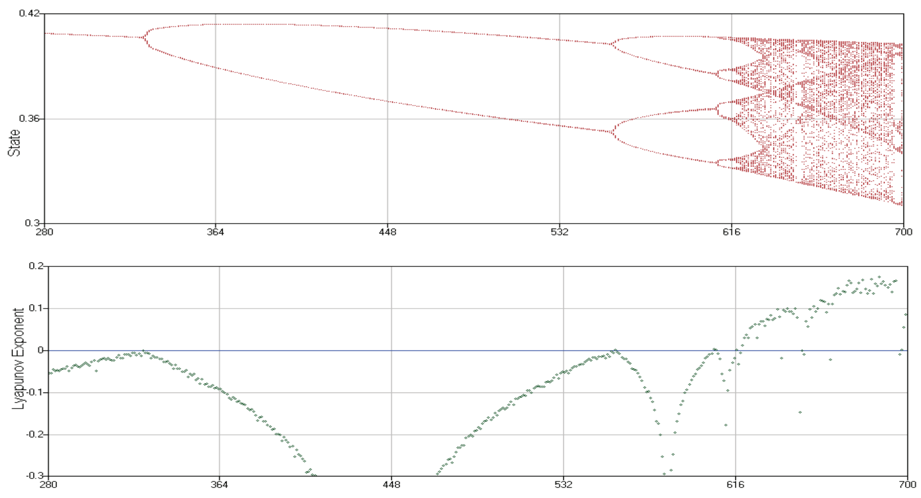
FIGURE 2.12. Hopf bifurcation for a 2-neurons recurrent neural network. Inputs (θ_0 and θ_1) on the network act as the control parameter. The corresponding Lyapunov exponents are plotted in green. The evolution function is $\forall i, j \in \{0, 1\} : x_i\left(t+1\right) = \theta_i + \omega_{ii}\left(\tanh\left(x_i\left(t\right)\right) + 1\right)/2 + \omega_{ij}\left(\tanh\left(x_j\left(t\right)\right) + 1\right)/2$.

unit circle, and as explained above the point of crossing determines the type of bifurcation the system undergoes. The evolution of eigenvalues gives indication on the nature of the road to chaos they describe [Albers et al., 1998]. Depending on this evolution, different kinds of chaos can be reached. Three of the most common scenarios are detailed below:

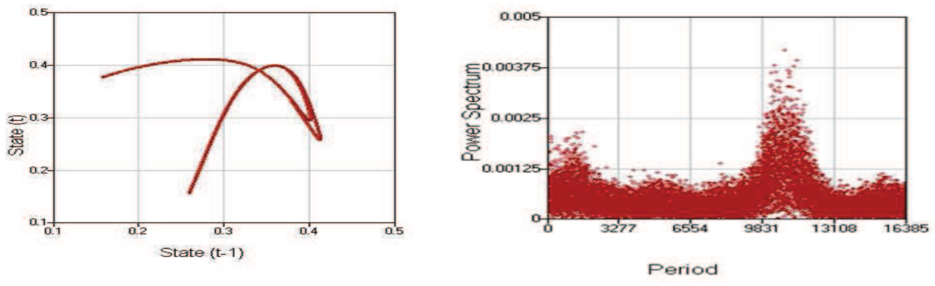
- **Road to chaos by period doubling**[71]: this is the most schematic road, occuring through a series of flip bifurcations. Feigenbaum [1983] reported that the period-doubling cascade of transitions on the road to chaos exhibits quite remarkable universal features: there are identical numbers characterizing the ratios of the control parameter for successive period doublings. Independently of the system, there seems to be universality in this kind of transition to chaos.
- **Road to chaos by quasi-periodicity**: First introduced by Ruelle and Takens [1971], this road occurs through successive Hopf bifurcations. Each bifurcation adds a new periodicity, starting with the tore $\mathcal{T}_1$ (one limit cycle), the system move to a tore $\mathcal{T}_2$ (two superimposed limit cycles), then to $\mathcal{T}_3$ (three superimposed limit cycles), etc. With each bifurcation, the dynamics becomes more and more complex and the system becomes more and more subject to resonances occurring between these frequencies. Such resonances appear when a rational factor occurs between different frequencies, hence when these frequencies tend to synchronize between each others. In such cases, the signals are also frequency-locked or phase-locked states.
- **The intermittency chaos**: To understand this road, it is important to first define the saturation of a variable and a system. A variable is saturated when its value reaches one of the extremum of its domain. For example, a real number variable cannot be saturated but a variable in the domain $]-1, 1[$ can be when $x \to \pm 1$. A system is saturated when its parameters are saturated; in this case a system with N parameter is limited to only 2^N possible states. When such a saturation occurs, the system stays on stable attractors. Chaos can appears if the evolution function is not completely saturated, which will be the case for most of the points in the trajectory: the function will be saturated but certain points will make one of the variables to become unsaturated and this will interrupt the periodic orbit before completion by an intermittent point, which is not one of the attracting points. After this miss, the system enters a chaotic phase until it falls back into the attractor at saturation of the squashing function.

All three types of road to chaos can be observed in neural networks; however the probability to see one vary with the way the network is built and with the network's architecture. Road to chaos by period doubling rapidly falls as the size of the network grows. The road to chaos by quasi-periodicity appears to be the most natural one for random recurrent networks of large size [see Dauce, 2000]. Both facts can be explained by the larger probability for the eigenvalues of the Jacobian matrix $\mathcal{D}F$ to be complex values, and hence Hopf bifurcations are much more frequent than flip bifurcations [Doyon et al., 1993]. As shown in Chapter 5 on page 101, iterative Hebbian procedures can be used as a road to intermittency chaos.

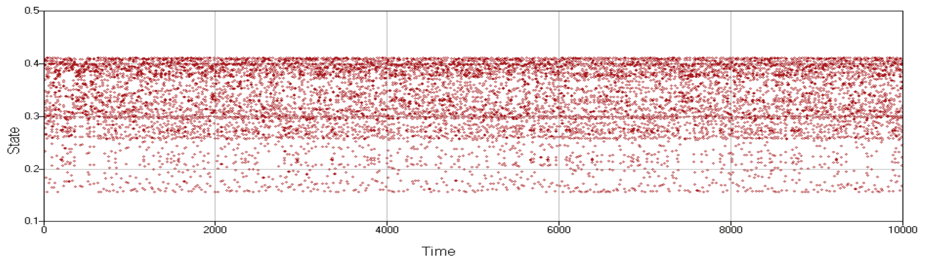
[71]Which is also known as the sub-harmonic road.



(a) Bifurcation Diagram

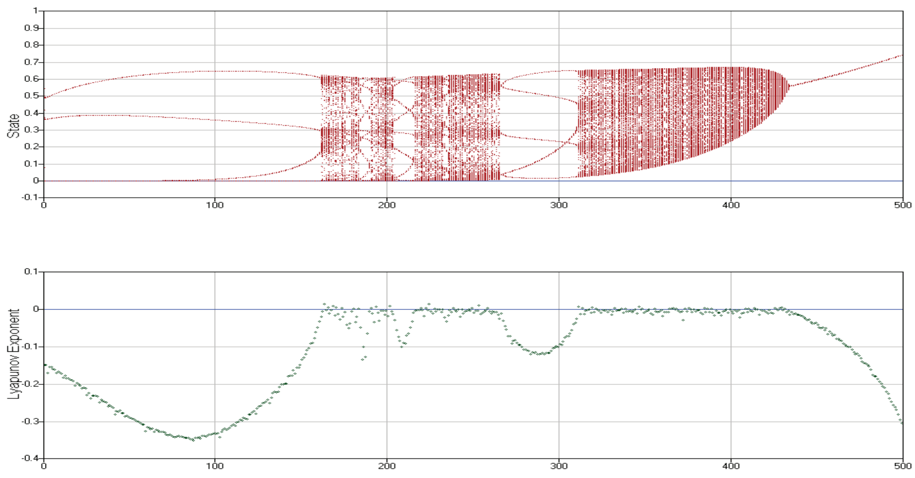


(b) Return Map and FFT for a chaotic region of the above diagram.

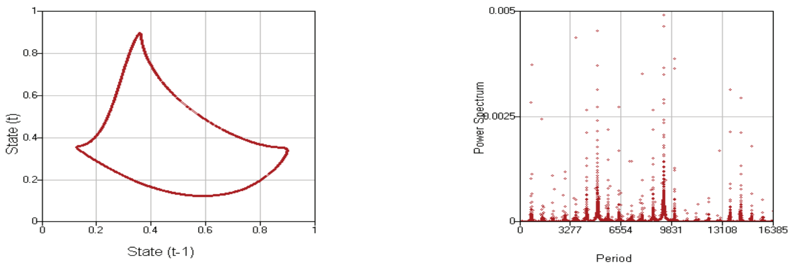


(c) State versus Time diagram.

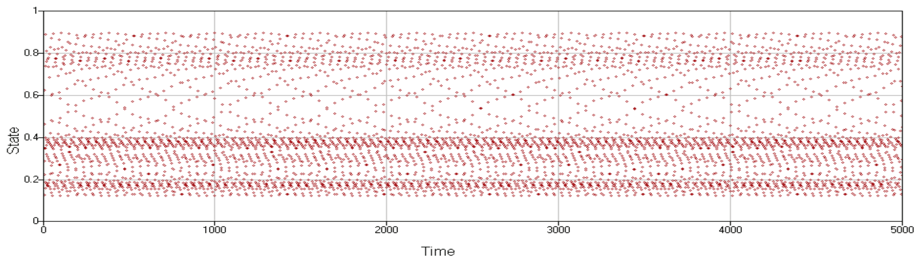
FIGURE 2.13. Period doubling road to Chaos and qualitative structure of this chaos for a 2-neurons RNN. The auto-connection weights of the network have been set to negative values while the weights between them are set positive. Inputs are used as control parameters.



(a) Bifurcation Diagram



(b) Return Map and FFT for a chaotic region of the above diagram.

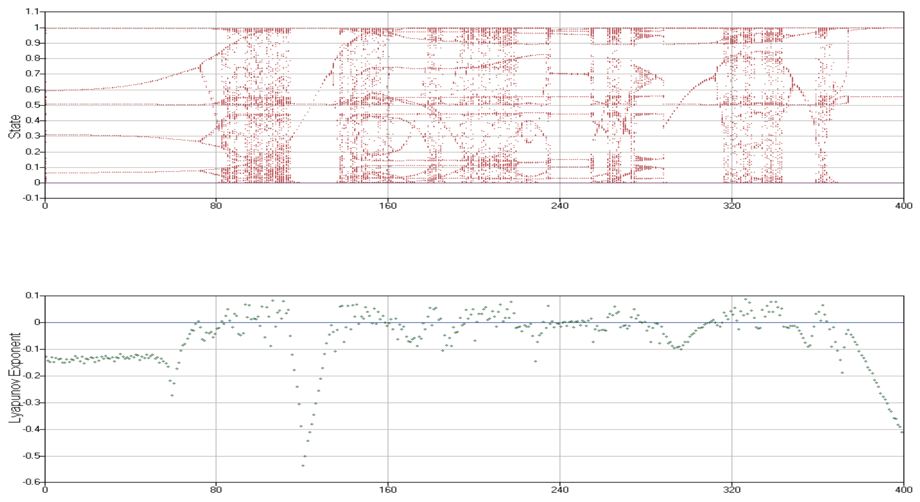


(c) State versus Time diagram.

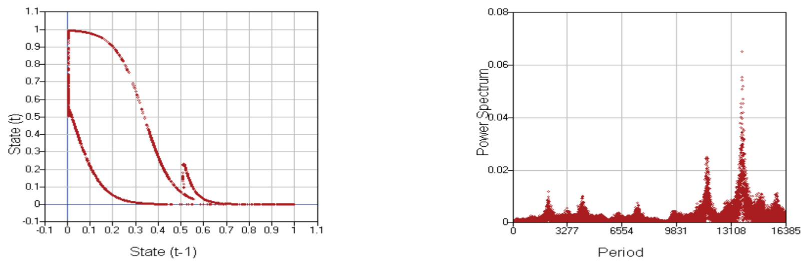
FIGURE 2.14. Quasi-periodic road to chaos and qualitative structure of this chaos for a 2-neurons RNN.

Figures 2.13 on the preceding page, 2.14, 2.15 on the next page show the three roads to chaos, respectively by period doubling, quasi-periodicity and intermittency. To further help to make explicit the differences between these roads to chaos a power spectrum, a return map and a state diagram[72] (for points located inside the chaotic region) have been added alongside the bifurcation diagram. One can note that in

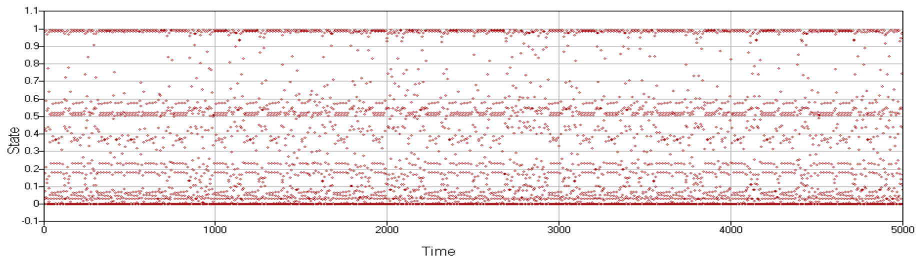
[72]See section 6.2 on page 48 for a detailed review of these tools.



(a) Bifurcation Diagram



(b) Return Map and FFT for a chaotic region of the above diagram.



(c) State versus Time diagram.

FIGURE 2.15. Intermittency Chaos and qualitative structure of this chaos for a 3-neuron RNN. Auto-connection weights are highly inhibitive.

the chaotic region of the parameter domain, all chaotic points show more or less the same structure (return maps and power spectra look very similar). However, when integrating information provided from the different plots it is often possible to determine the type of chaos the system is going through.

However, with real size neural networks the parameter space becomes rapidly beyond control[73] which makes it difficult to obtain a bifurcation diagram that shows a perfect road to chaos and that is in perfect correspondence with the theory exposed above. To plot a bifurcation diagram an intersection has to be made in the high-dimensional parameter space with a one-dimensional space (the other dimension of the parametric change). This makes finding the point of the first bifurcation very difficult.

6.4. Dynamical System Theory & Recurrent Neural Networks. Neural networks are perfect candidates to be analytically studied with the bifurcation theory: they are complex dynamical systems with the presence of nonlinear interactions between the units. However, the parameter space grows rapidly with the size of the network, making such an approach very unlikely.

6.4.1. *The one neuron recurrent neural network.* The self-interacting one neuron network was solved by Pasemann [1993] more than a decade ago. Given the following evolution equation:

$$x\left(t+1\right)=\theta+\omega f\left(x\left(t\right)\right) \tag{17}$$

where θ is the input on the neuron and f is the threshold function, here a simple sigmoid function. Three different types of trajectories can be found in three different domains in the (θ,ω)-parameter space (see Figure 2.16). The boundaries between these domains represent the set of values in the parameter space leading to non-hyperbolic points. At these points, the norm of the eigenvalue of the Jacobian is equal to one.

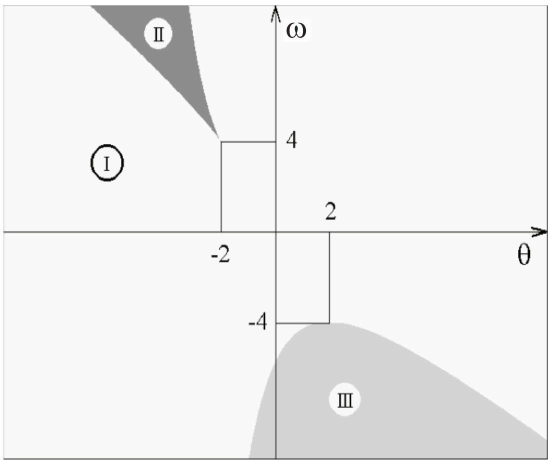


FIGURE 2.16. One neuron's dynamics, shown within the (θ,ω)-parameter domains. It shows global fixed point attractors (I), bistability (hysteresis)(II), and global period-2 orbit attractors (III). The evolution function is $x\left(t+1\right)=\theta+\omega f\left(x\left(t\right)\right)$

For a self-excitatory neuron ($\omega>0$), a hysteresis domain (II) exists over which the system has two coexisting fixed point attractors. This domain is bound by a bifurcation set, which is determined by a cusp catastrophe at $(\theta_c,\omega_c)=(-2,4)$. For a self-inhibitory neuron ($\omega<0$), there is an oscillatory domain (III) corresponding to global period-2 orbit attractors. It starts at the critical point $(\theta_c,\omega_c)=(2,-4)$. Parameter values outside these two domains have a global fixed-point attractor (I).

[73]The parameter space in $O\left(n^2\right)$.

Figure 2.11 on page 54 actually shows the domain (II) on the left plot and the domain (III) on the right plot.

This system does not exhibit any kind of chaotic dynamics. Pasemann [1997] successfully obtained chaotic dynamics from this model by adding a linear term or persistence:

$$x\left(t+1\right)=\gamma x\left(t\right)+\theta+\omega f\left(x\left(t\right)\right) \tag{18}$$

where γ is a dissipative parameter.

6.4.2. *The two neurons recurrent neural network.* The two neuron system has also been completely solved analytically by Pasemann [1999]. This is the simplest neural network which exhibits all kinds of non trivial dynamics, ranging from stationary attractor to oscillation of various periodicity and even quasi-periodicity and chaotic dynamics (without any addition, such as the persistence state in Equation 18).

The system can be formally written as the following map: $F_\rho\colon \mathbb{R}^2 \to \mathbb{R}^2$, where $\rho = (\theta_0, \theta_1, \omega_{00}, \omega_{01}, \omega_{10}, \omega_{11}) \in \mathbb{R}^6$:

$$\begin{array}{rcl} x_0\left(t+1\right) & = & \theta_0+\omega_{00}f\left(x_0\left(t\right)\right)+\omega_{01}f\left(x_1\left(t\right)\right)\\ x_1\left(t+1\right) & = & \theta_1+\omega_{10}f\left(x_0\left(t\right)\right)+\omega_{11}f\left(x_1\left(t\right)\right)\end{array} \tag{19}$$

The equation giving the boundaries of the multiple domains are far from trivial to solve and even harder to analyze. Even representing the parameter space requires compromise since it is now six dimensional. However, for specific parameter configurations (approaching the Hopf bifurcation) the output signal of neurons is almost sinusoidal, and its frequency can be controlled by a single parameter [Pasemann et al., 2003]

6.4.3. *Larger recurrent neural network.* No complete theoretical analysis exists for recurrent neural networks larger than two neurons. Even if such a study was possible, it would be extremely complicated since equations become very difficult to solve and the number of parameters rises quadratically with the number of neurons. The reason why this study has lost any interest for many researchers is that the complexity is such that even if an analytic solution was found it seems unlikely that it would help get a better understanding of the system, yet alone provide practical use for these networks.

Another perspective is to analyze large connectionist networks with the use of statistical physics. Following the leads of Hopfield [1982] who proposed an analogy between neural networks and glass spin networks, or as suggested by Amari [1983], at the thermodynamic limit (when size tends to infinity), it is possible to identify a limited set of global observables which characterize some properties of the system. Here, the neural network is seen as an emergent system, where some global properties (which can be mathematically described) emerge from its individual neurons. An intermediate solution is to work at the mesoscopic level, where the collective behavior of a finite number of neurons is considered [Cessac, 2002].

6.5. Hysteresis. To understand hysteresis, it is important to understand the notion of initial condition and basin of attraction; these two notions play a crucial role in dynamical systems. Here follows, a non formal reminder of those two concepts:

initial condition: is the starting N-dimensional point in space where the trajectory starts it revolution. In a neural network, the initial inner state vector gives this.

basin of attraction: is a set of points in the state space which converge to an attractor.

In the one-neuron study, it has been shown that (depending on the parametric configuration) two basins of attraction coexist. Depending on its initial condition the system will end in one or the other. In higher-level systems (larger neural networks), multiple dynamics can coexist at the same time. Depending on the initial conditions, the system evolves in a limit cycle, a quasi-periodic limit cycle or a strange attractor. It has to be noted that the initial Hopfield model developed to store memories is based on the coexistence of different attractors.

hysteresis phenomenon: appears when the system evolves through control parameters and crosses a region of multi-stability where different basins of attraction coexist. This phenomenon is observed when a control parameter is changed from one value to another and then brought back to its original value. The difference between the two parametric changes is the initial conditions and the direction of movement but they both cover the same parametric space. The forward pass and the backward pass result in different bifurcation diagrams (called "path-bifurcation diagram").

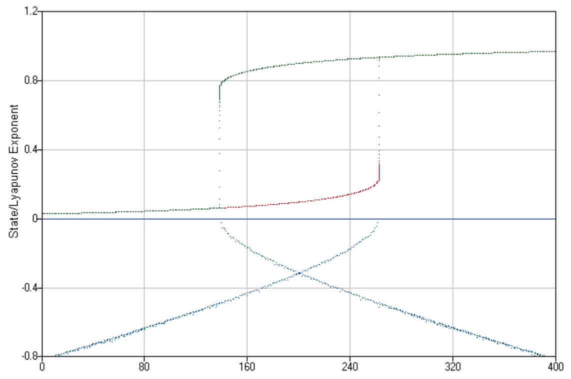


FIGURE 2.17. Path-bifurcation for the one-neuron recurrent neural network. A simple hysteresis appears and corresponds with a bi-stable domain. The Lyapunov exponent is equal to 0 when saddle-node bifurcations occur.

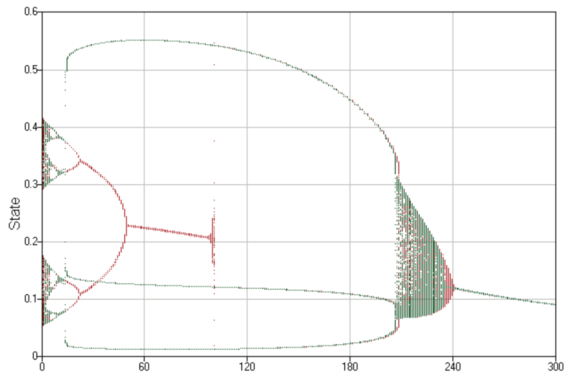


FIGURE 2.18. Path-bifurcation diagram for a 3-neurons recurrent neural network. It shows a complex hysteresis phenomenon.

Figure 2.17 on the preceding page shows the hysteresis phenomenon occurring in a one-neuron recurrent neural network while the input θ is changed. This hysteresis results from the existence of a domain of bi-stability. Figure 2.18 on the previous page shows a more complex hysteresis phenomenon appearing in a larger recurrent neural network.

This phenomenon shows the importance of initial conditions. Furthermore, it shows how they can lead to different bifurcation diagrams. A bifurcation is said to be global if it occurs independently from the initial condition, the bifurcation is called local in other cases. It has to be noted that the first point of bifurcation (the first eigenvalue crossing the unit circle) is always a global bifurcation [Albers et al., 1998]. Before this bifurcation, there is only one attractor (a fixed point), and after it, dynamics are often not global.

Pasemann [1997] has suggested the use of this phenomenon in the one-neuron recurrent neural network as building blocks of a short-term memory. He also demonstrated that hysteresis effects can improve performance for robots trained to support navigation tasks [Huelse and Pasemann, 2002].

7. Conclusion

This chapter introduced the fundamental notions, which will serve as a basis to this thesis. Being in the cross section of several fields this chapter had to go through lots of different concepts that one may not at first fit together, even if in the end they all form a coherent ensemble in this work. The first important field to cover was the neurophysiology whose goal was to familiarize the reader with the current understanding, facts and hypotheses of the brain study. One particular construct which has been looked at a bit more closely is the memory which is at the heart of this work; this in turn required a more detailed definition of one region of the brain: the hippocampus. This part of the chapter covered the biological basis of the work, which will serve as guideline through the work.

Since this thesis is dedicated to computational models and works with artificial neural network, the corresponding modeling was introduced as the connectionist model and compared to the alternative computationalist model. The numerous models of artificial recurrent neural networks and the lack of convincing applications for these networks clearly depict the state of the art in this field. These recurrent networks seem so powerful that they appear difficult to control and to learn. It has been showed that they are extremely hard to study by analytic approaches and that results are very rare when the network's size goes over two neurons, while practical cases require the use of hundreds, even thousands of neurons which in turn is still far from the biological scale of billions of neurons.

The final piece in this puzzle is the complex system theory. From biological evidences and studies to engineerist models, everywhere complex dynamics seems to be part of the problem, and part of the solution. It was therefore important to introduce properly the basic notions of the nonlinear dynamical systems, and more specifically the theories of bifurcation and chaos. Connectionist models have also proven to be connected deeply to the complex dynamics, especially when dealing with neural networks used as memory. The correlation between the presence of chaos and the capacitive limits of a neural network has been shown by Molter and Bersini [2003a,b], this is covered in more detail in the Chapter 4 on page 81.

CHAPTER 3

Model

As explained in the previous chapter, neural networks can be modeled in lots of different ways; they suggest that the study of mental activity is really the study of neural systems. This links connectionism to neuroscience, and models involve varying degrees of biological realism. They do not need, in general, to be biologically realistic. Those models range from very simplistic definitions (i.e.: modeled as a graph) to very complicated systems (i.e.: modeling the dendrites, the soma, the synapses, ...), which mimic aspects of natural neural systems very closely[1]. A neural network model definition is based upon several important building blocks: it needs a proper definition of the neurons, the way those neurons are connected together, how they evolve through time, how time is represented in the model, etc.

The complexity that comes with the design directly impacts on the performance of the system as a whole. The goal of this chapter is to provide the simplest possible definition that is biologically plausible and may be used to give a better understanding of the real systems being modeled. This chapter presents such a model, and validates the choices that come with this particular model. First, it introduces existing related approaches and highlights the methodology behind the construction of the model used in this work.

1. Methodology and State of the art

1.1. Related approaches.

Freeman and Kozma et al.: Freeman has done neurophysiological experiments over 20 years, showing evidence of complex activities occurring in the brain through observations of EEG signals obtained from different brain locations [Skarda and Freeman, 1987].

Later, Kozma and Freeman [2001] proposed to build an artificial model replicating at best brain dynamics. They use simple building blocks that were put together into more complex architectures creating new super building blocks, which in turn can be used to create even more complex architectures. Muthu et al. [2004] describe this as:

> "K sets" represent a family of models of increasing complexity. Each level of complexity represents various aspects of operation of vertebrate brains. These models are biologically inspired, and they are built based on the salamander's central nervous system. They provide a biologically conceivable simulation of chaotic spatio-temporal neural developments at the mesoscopic and macroscopic scale

For example, the K-IV model (the last one in the chain) consists of four components: the hippocampus (for the spatial orientation), the cortical

[1]The so-called "neuromorphic networks".

region (for sensory inputs), the midline forebrain (deals with the internal goals) and the amygdala (aids the system for decision-making while navigating). All parts are modeled with a specific K-III set.

This methodology had two main goals: first learn about the neural assemblies and second develop autonomous robots capable of performing cognitive tasks.

Doyon and Samuelides et al.: This group of researchers, starting in the 1990s, have used mathematical approaches to extend the thermodynamic limit (when the network's size tends to infinity) on the studies of the dynamical behaviors of these models [Doyon et al., 1993, 1994]. Their work takes its root in the work of Amari [1983], who showed how a model formed by a huge number of microscopic variables can be described using some macroscopic observables. Interestingly, they have studied intensively one particular two-neuron network, where one neuron is inhibitory and the other is excitatory; and this is nothing more than the K-II model proposed by Kozma and Freeman.

Later more work has been done to use this knowledge in autonomous robots. While, Freeman *et al.* tried to obtain cognitive behaviors by "creating an artificial brain", Dauce et al. [1998] tried to obtain the same cognitive behaviors by finding Hebbian-based learning procedures.

Pasemann et al.: During this period, Pasemann instigated another neurodynamics approach. He obtained theoretical results on the one-neuron network [Pasemann, 1993] and the two-neuron network [Pasemann, 1999] and even applications were proposed based on these results. One such application was the development of the one-neuron chaotic network [Pasemann, 1997], another was the creation of a generator of frequencies using a two neurons network [Pasemann et al., 2003]. These results were also compared (qualitatively) with brain dynamics [Pasemann, 2002].

Again, like the two previous groups, he tried to apply these results into autonomous robotics. Nevertheless, even if the goal was the same, the means were different. Each group gave a different response to the following question: "what type of recurrent structure is to be used for the generation of a successful behavior?". Freeman *et al.* created a complex architecture (the K-IV model), Doyon *et al.* used a simple model of huge size, where Huelse and Pasemann [2002] developed an evolutionary algorithm for the structural development of neural networks that use a fitness function, called ENS[2]. One interesting result reported by these researchers is that most often well skilled robot behaviors are partly due to hysteresis effects associated to specific recurrences [Huelse and Pasemann, 2002]. Later, Harter and Kozma [2005] conducted comparative studies with the Huelse and Pasemann [2002] model concerning the robustness and memory capacity.

Carpenter, Grossberg and Kohonen: Carpenter and Grossberg [1988] have proposed the "Adaptive Resonance Theory", simply known as ART. From their first proposed model many evolutions and improvements have been suggested [Carpenter and Grossberg, 2003]. ART is a system capable of online-unsupervised learning of new memory. The basic idea of the ART systems is a pattern matching process that compares an external stimulus with the internal memory. This matching leads to a parallel search or to a *resonant* state, which persists long enough to permit learning. The search may end at an established code, which allows the existing

[2]ENS means "evolution of neural systems by stochastic synthesis".

memory representation to be altered; or it ends at a new code, and the memory representation learns the current input. This allows the system to autonomously either learn a completely new pattern or adjust an existing one.

In the same vein, the "Self-Organizing Map" (SOM) suggested by Kohonen [1998] is another autonomous system that learns through unsupervised learning. The goal of SOMs is to map a high-dimensional distribution onto a regular low-dimensional grid. This allows seeing a complex nonlinear statistical relationship between high-dimensional data items as a simple geometric relationship on a low-dimensional display. Even if it may not seem very close to a memory model, the learning in SOMs works by causing different parts of the network to respond similarly to certain input patterns. The SOMs have some biological basis related to how separate parts of the cerebral cortex in the human brain handle visual, auditory or other sensory information. Unfortunately, this model lacks biological likelihood in its implementation.

1.2. Approach of this thesis. There is of course a lot in common amongst what will be presented here and the work of the previously cited authors [Guillot and Dauce, 2002; Skarda and Freeman, 1990b]. There is an agreement that dynamical signals present in the brain play a big part in the information present in it. In addition, neurophysiological observations show the importance of the environment in the learning process: learning mechanisms involve ongoing adaptations of the brain throughout its lifetime in response to the environment. These observations raised the idea that it would not be possible to consider any cognitive process by dissociating the connectionist model from its environment [Brooks, 1986]. However, most of those groups have put their model in use in autonomous robotics, and analyzed the impact of learning mechanisms in the network's underlying dynamics in conjunction with the observed behavior. Here, the models are studied following the idea that information is stored in the brain in almost cyclic dynamics while attentive waiting states correspond to chaotic dynamics, and thus the artificial recurrent neural networks are used as memory. It is also clear that to be able to produce a brain-like memory, the system needs to be able to learn autonomously and unsupervisedly. This is where the works from Carpenter and Grossberg [1988] and Kohonen [1998] are very close to the goal of this thesis even if their models have a biologically unlikely implementation.

The following chapters will show what kind of results can be obtained and what biological precision they carry with them. Even when the model's working hypotheses are not always biologically valid, they allow highlighting some important properties that can go beyond those hypotheses. Since the main topics of this work is about memory and information storing, it is important to give a proper meaning to the term "information" in this work. The brain processes information in many ways, and like the brain, neural networks can be seen as black boxes, which receive information (or data) through their input neurons and, given some personal history (the network internal state), process this information and then act, this "action" being observed on their output neurons.

The feed forward neural networks can associate a given input to an output, changing its syntactic and/or semantic meaning in the process. This is a hetero-associative way of associating information. These networks have been widely studied for clustering and classification purposes [see Haykin, 1998, amongst others]. If some internal recurrences are added, the output is no longer uniquely determined by the input. The network internal dynamics also affects the generated output action. This family of feed forward recurrent networks [Kremer, 2001] can have lots of different architectures. A typical application of these models is the simulation of context sensitive dependencies required for language processing.

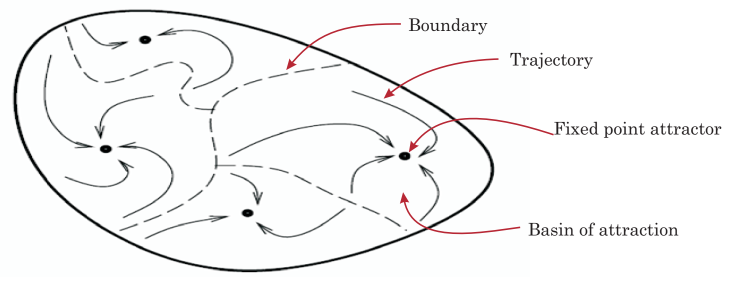


FIGURE 3.1. Basins of attraction in the state space with their corresponding fixed-point attractors. Information is mapped into a fixed-point attractor, which defines the system as an auto-associative memory (or memory addressable by contents), since partial information will be attracted to the closest attractor.

The models proposed by Hopfield [1982], changed things in the connectionist models[3] He developed a new kind of architecture, the fully recurrent neural network, where inputs are no longer seen as external to the system: they directly initialize the internal states of the neurons, which forces the network into an initial location of its state space, which can be related to the vicinity of an attractor. Furthermore, by constraining the weights of the connections to be symmetric, the network is assured to converge to fixed points attractors. Furthermore, Hopfield found a learning algorithm based on the Hebbian prescription satisfying this weight constraint and enabling the association of information with fixed points attractors of the dynamics. This model defines an auto-associative way to store information: the information in the input is the same as the information in the output and the existence of basins of attraction permits recovery from noisy inputs (see Figure 3.1). The main difference between auto-associative and hetero-associative memory is the way the information is addressed. Auto-associative memory allows addressing the information by contents, while hetero-associative memory is closer to computer-like memory addressed by address.

Following, this idea, coupled with the motivation to use complex dynamical systems as part of the encoding scheme, the work presented here builds on top of models like Hopfield's but focuses on using richer dynamics to do the bidding. It is possible to go even further and work locally on the network. In Hopfield's model the whole system gives the answer and the whole system converges to a given dynamics. As it seems unlikely to have fixed-point attractors in the brain, it is even more unlikely for these attractors to be global in the system. It is also obvious that such systems will scale poorly with the size of the network.

One possible way to represent information is through cell assemblies as introduced by Hebb [1949]. From what is explained in section 1.4 on page 14, a cell assembly can be seen as a group of neurons that react in a synchronous way to a given input. It is important to note that those groups can overlap (partially or even totally). Here "synchronous way" has also a specific meaning: since this thesis does not work with continuous time models the synchrony of two signals is less obvious

[3]At the same time, Cohen and Grossberg [1982] also developed a more general approach which contributed to these changes.

to define: there are different ways this can be extended to the discrete time systems nonetheless. The simplest way is to see any two units that share similar activation patterns during a period of time to be synchronous. Another way is to look at the average activity of two units (neurons) over a period of time T and if both are above a threshold they are considered synchronously active. As one may note, this is a much more loose definition of synchrony, compared to two continuous oscillatory signals having the same phase and frequency.

2. Unit Definition

A neural network can be seen, in its simplest form, as a weighted directed graph. From the biological point of view, here each node represents a neuron and each edge represents the synapse between the two neurons. The following section properly defines the node and the edges of this graph.

2.1. Neurons. A neural network $\mathfrak{A}$ is composed of N neurons, $|\mathfrak{A}| = N$ indicates the size of the network and $i \in \mathfrak{A}$ denotes the ith neuron of $\mathfrak{A}$.

The simplest way to model a neuron is a numerical value that evolves thought time, noted $x_i(t)$, where i is the neuron. This numerical value can either be the current activity of the neuron (the intensity of the action potential itself) or this value can just indicate the rate at which the neuron is firing (and not the spike's actual intensity). Usually, when modeling the neuron's spiking activity, one needs to have to compute the system using a continuous time scale ($t \in \mathbb{R}^+$). On the other hand, when dealing with rate firing models, discrete time scales are also an option ($t \in \mathbb{N}$). The value of the neuron is also called "inner state" or just "state".

The neuron's state can also be a discrete ($x_i(t) \in \mathbb{Z}$) or a continuous ($x_i(t) \in \mathbb{R}$) value, with any kind of domain. Common domains for discrete state neuron are he binary domains $\{0, 1\}$, $\{-1, 1\}$ and symbolic domains $\{a_0, a_1, \ldots, a_n\}$ (e.g.: music notes). For the continuous state neurons the most common domains are $[0, 1]$ and $[-1, 1]$. To compare the network's continuous internal states with bit-patterns, a filter layer quantizing the internal states, based on the sgn function, can be added [Omlin, 2001]. It defines the output vector $\mathbf{o}$:

$$\begin{cases} o_i = -1 & \Longleftrightarrow \quad x_i \quad < 0 \\ o_i = 1 & \Longleftrightarrow \quad x_i \quad \geq 0 \end{cases} \tag{20}$$

where x_i is the internal state of the neuron i and o_i is its associated output (i.e. its visible value). This filter layer enables to perform symbolic investigations on the dynamical attractors.

Figure 3.2 represents a period 2 sequence unfolding in a network of four neurons. The persistent external stimulus feeding the network appears in the upper part (A) of the figure. Given that the internal state of neurons is continuous, the internal states (B) are filtered (C) to enable the comparison with the stored data.

Of course, neuron's state value can be in any arbitrary set but those models tend to add a layer of complexity, which is not computationally efficient. Common solutions include $x_i(t) \in \mathbb{C}$ or $x_i(t) \in \mathbb{R}^n$ (where n is a priori defined). Those multi-valued neurons are a way to have a spiking model without using continuous time. The easiest example is the complex number: when presented in polar form the two values of a neuron are ρ and Θ where ρ represent the amplitude of the spike, and $\Delta\Theta$ represents the firing rate (frequency).

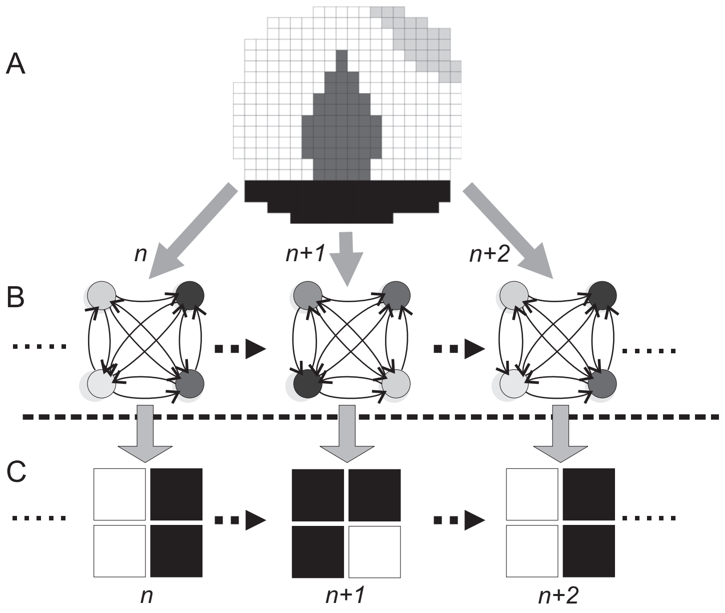


FIGURE 3.2. Picture of a fully recurrent neural network (N=4) fed by a persistent external stimulus (A). Three snapshots of the network's states are shown (B). Each one represents the internal state of the network at a successive time step. After filtering, a cycle of period 2 can be seen (C). The input layer is fully connected to the associative layer.

2.2. Connections. Connections between neurons are represented as directed weighted edges on the graph; when working with fully connected recurrent neural networks, the best way to model the connections is through a dense weight matrix. This matrix is usually denoted $\mathbf{W}$, and an individual value of this matrix is denoted w_{ij}, which denotes the weight of the edge going from the neuron j to the neuron i. Each line i of this matrix represents all the incoming connections from all neurons to the neuron i.

As with the inner states, usually the weights are real numbers. The most common approaches are $w_{ij} \in \mathbb{R}$ or $w_{ij} \in \mathbb{R}^+$. A positive weight value ($w_{ij} > 0$) indicate an excitatory neuron (j excites i) and in a similar way a negative weight value ($w_{ij} < 0$) indicates an inhibitory neuron (j inhibits i). Biological evidences suggest that a neuron can only have one role (excitatory or inhibitory). The next section provides a proof, that a model where neurons can have multiple roles ($\exists i_1, i_2 \ \operatorname{sgn}(w_{i_1 j}) \neq \operatorname{sgn}(w_{i_2 j})$) is equivalent to a model with single roles ($\forall i_1, i_2 \ \operatorname{sgn}(w_{i_1 j}) = \operatorname{sgn}(w_{i_2 j})$) .

More complex systems exist where $w_{ij} \in \mathbb{R}^n$ for instance. One such system can be, for example, a model where the weights are seen as twofold: first the permanent weight (w_{ij}^p) between two neurons and secondly, the facilitation weight (w_{ij}^f) which is a bias to the permanent weight and can help accentuate it or dampen it. Commonly the final weight on the edge is $w_{ij} = w_{ij}^p + w_{ij}^f$ or $w_{ij} = w_{ij}^p w_{ij}^f$. But again, this kind of definition can severely affect the performance of the system, since the final value of the weight always requires some form of computation and the system needs also to take care of the facilitation weight as a transient value which requires some more computation each time the system evolves.

2.3. Synchronous vs. Asynchronous. An important component, when simulating neural networks, is synchrony. Either all neurons can be simulated at once in a synchronous way or each neuron can be simulated in an arbitrary order[4] in an asynchronous way. There is a strong impact of this choice over the dynamics that can be expected in the system. For example, Hopfield networks are guaranteed to converge to a fixed-point dynamics for any starting point if simulated in an asynchronous way. A contrario, it has been shown that those networks will have cyclic (or more complex) dynamics if simulated synchronously.

The model presented here is in fact very close to Hopfield networks for many aspects, and synchronous simulation is needed to get complex dynamics in the system[5]. Fixed-point attractors are a good property for physical systems but, when modeling the brain, the evidences suggest that they do not seem plausible in a normal brain activity.

2.4. Simulation. The simulation is the most critical part of the neural network model: it describes how the inner states of the neurons evolve through time. The complexity of the simulation function is correlated with the choices made when modeling the neuron and connections of the neural network. For example, simulating a system in discrete time scale is much easier than in a continuous time scale. A simulation function (also called "activation function") needs to specify how the model reflects the inter-neuron activity. The most common way to do this is to compute for each neuron a weighted sum of the incoming inputs. This can be formalized as[6]

$$x_i\left(t\right) = f\left(\sum_{j=1}^{N} w_{ij} x_j\left(t-1\right)\right) \tag{21}$$

where:

- x_i represents unit i's firing rate;
- w_{ij}, the synaptic weight connecting cell j to cell i;
- f, the threshold function, depends on the model and can even be different for separate neuronal groups in a given model (f may be replaced by f_i).

This formulation is known as McCulloch and Pitts neurons evolving in discrete time step [McCulloch and Pitts, 1943]. It describes a synchronous simulation since $x_i\left(t\right)$ depends only on $x_j\left(t-1\right)$, which all come from the previous time step. Since there isn't any constraint on the weights: $\sum_{j=1}^{N} w_{ij} x_j\left(t-1\right) \in \mathbb{R}$. This is why the threshold function was chosen to map the weighted sum to the desired domain of the neurons' inner state. The most common threshold function is $f = \tanh\left(x\right)$ which maps $\mathbb{R} \to \left]-1, 1\right[$.

When modeling a rate firing model, extra care needs to be taken for the threshold function. If the inner states of the neuron represent the firing rate of the neuron then it must always be a positive value (e.g. $f : \mathbb{R} \to [0, 1]$). The neuron's firing

[4]This order does not need to be specified beforehand and can be implemented in different ways. The goal here is to mimic the independent behavior of neurons where each one fires at its own rate. Common approaches use a predefined simulation order, random order at each simulation step or a probabilistic simulation where each neuron has a chance to be the next one to be simulated.

[5]Hopfield showed that the system needed asynchronous simulation amongst some constraints on the weights matrix in order to guarantee to always stabilize in fixed-point attractor.

[6]continuous time scale models are describe as a derivative of x over time : $\dot{x}\left(t\right) = \dots$

rate needs to be null in absence of inputs. Here the following function[7] is proposed for the rate firing model[8] (see Figure 3.3):

$$f(x) = \frac{\tanh(3x-2)+1}{2} \tag{22}$$

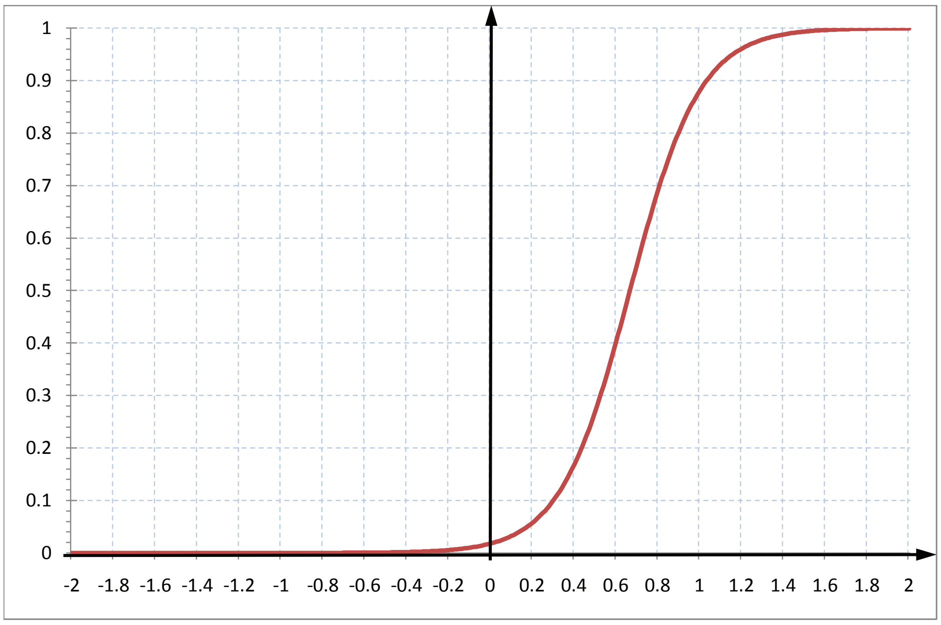


FIGURE 3.3. Threshold function of the rate firing model used in this work, described by $f(x) = \dfrac{\tanh(3x-2)+1}{2}$

Historically, the step function was chosen [McCulloch and Pitts, 1943] as a threshold function; here some kind of sigmoid function has been preferred. The reason for this is pretty simple: when dealing with a sigmoid-like function it is always possible to later look at the output of a neuron through a filter which will quantize it ($[0,1] \rightarrow \{a_0, a_1, \cdots, a_n\}$). On the other hand a continuous output enables the computation of all kinds of properties over the dynamics of the system (e.g. a Lyapunov exponent).

2.5. Matrix form definition. Since a synchronous discrete time simulation is used and the activation function is a weighted sum, the natural way to represent the system and the simulation is through vector, matrix, and matrix vector product. A proper definition for the states of the network, the weights, and the activation function is needed.

The inner state of the neurons is defined as the vector whose components represent each neuron's inner state (defined as $\mathbf{X}(t) = [x_0(t) \;\; x_1(t) \;\; \ldots \;\; x_N(t)]$, where

[7]The given function does not exactly satisfy $f(0) = 0$, but is close enough and can be taken arbitrarily close to zero by choosing a constant $K > 2$ and redefining the function as $f(x) = \dfrac{\tanh(3x-K)+1}{2}$. This function also maps $\mathbb{R}$ to $]0,1[$ but when dealing with finite precision real numbers (as with computers) it is safe to assume the set to be closed.

[8]The model does not implement any form of short-term memory feature at the cell level.

N is the size of the network). The weight matrix has already been defined as $\mathbf{W}$ (in the section 2.2 on page 68). The activation function can be now redefined as:

$$\mathbf{X}(t) = F(\mathbf{W}\mathbf{X}(t-1)) \tag{23}$$

$F(\mathbf{X})$ is the generalization of the threshold function working on vectors. It is defined as:

$$F\colon \mathbb{R}^N \to \mathbb{J}^N \subseteq \mathbb{R}^N : \mathbf{X} \mapsto F(\mathbf{X}) = [f(x_0)\, f(x_1) \dots f(x_N)] \tag{24}$$

Where $f : \mathbb{R} \to \mathbb{J}$ is the threshold function (e.g.: for rate firing model $\mathbb{J} = [0, 1]$, see eq. 22 on the preceding page)

This matrix representation allows fast network simulations by relying on libraries such as BLAS[9] and LAPACK[10].

3. Equivalence Theorem

As discussed in section 2.2 on page 68, the defined model does not satisfy the biological evidence that neurons can only be excitatory or inhibitory. However, this thesis shows that it is always equivalent to a model which does satisfy this criteria. The idea behind this proof is to show that it is always possible to construct a neural network which respects the biological connectivity constraint and which is equivalent to the original network for all of its initial conditions. To this end, this section will first formally define the equivalence between two networks.

Definition 7. *A neural network $\mathfrak{B}$ is equivalent to a network $\mathfrak{A}$ if there is a relation $\mathfrak{R} \subset \mathfrak{A} \times \mathfrak{B}$ with $\mathfrak{R}(\mathfrak{A}) = \mathfrak{B}, \mathfrak{R}^{-1}(\mathfrak{B}) = \mathfrak{A}$ and $\left(\forall (i,j) \in \mathfrak{R}\colon x_i^{\mathfrak{A}}(0) = x_j^{\mathfrak{B}}(0)\right) \Rightarrow \left(\forall (i,j) \in \mathfrak{R}, \forall t \in \mathbb{N}\colon x_i^{\mathfrak{A}}(t) = x_j^{\mathfrak{B}}(t)\right)$*

Given this definition, the equivalence theorem says that for any neural network $\mathfrak{A}$ where a neuron can have both excitatory and inhibitory connections, it is possible to construct an equivalent neural network $\mathfrak{B}$ which do not possess any neuron with both inhibitory and excitatory connections, with the same orbits for the same initial conditions.

Theorem 3.1. *Given a neural network $\mathfrak{A}$ of size N with arbitrary weights $\mathbf{W}^{\mathfrak{A}}$ and inner states $\mathbf{X}^{\mathfrak{A}}$, there is another neural network $\mathfrak{B}$ equivalent to $\mathfrak{A}$ that satisfies $\forall j\colon \forall i,k\colon \operatorname{sgn}\left(w_{ij}^{\mathfrak{B}}\right) = \operatorname{sgn}\left(w_{kj}^{\mathfrak{B}}\right)$.*

Proof. The weight matrix of network $\mathfrak{B}$ of size $M = 2N$ can be defined from $\mathbf{W}^{\mathfrak{A}}$ as:

$$\begin{aligned} w^{\mathfrak{B}}_{(2i)(2j)} = w^{\mathfrak{B}}_{(2i+1)(2j)} &= \begin{cases} w^{\mathfrak{A}}_{ij} & \text{if } \operatorname{sgn}\left(w^{\mathfrak{A}}_{ij}\right) > 0 \\ 0 & \text{otherwise} \end{cases} \\ w^{\mathfrak{B}}_{(2i)(2j+1)} = w^{\mathfrak{B}}_{(2i+1)(2j+1)} &= \begin{cases} w^{\mathfrak{A}}_{ij} & \text{if } \operatorname{sgn}\left(w^{\mathfrak{A}}_{ij}\right) < 0 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{25}$$

The mapping relation $\mathfrak{R}$ is defined as $\mathfrak{R} = \mathfrak{F}^{-1}$, where $\mathfrak{F}$ is the following surjective function:

[9] http://www.netlib.org/blas/
[10] http://www.netlib.org/lapack/

$$\mathfrak{F}\colon \mathfrak{B} \to \mathfrak{A}\colon \mathfrak{F}(i) = \left\lfloor \frac{i}{2} \right\rfloor \tag{26}$$

This defines the initial conditions of $\mathfrak{B}$ from those of $\mathfrak{A}$ as:

$$x_{2i}^{\mathfrak{B}}(0) = x_{2i+1}^{\mathfrak{B}}(0) = x_i^{\mathfrak{A}}(0) \tag{27}$$

By construction, it is trivial that:

$$\begin{array}{rcl} \forall i,j\colon \operatorname{sgn}\left(w_{i,2j}^{\mathfrak{B}}\right) & \geq & 0 \\ \forall i,j\colon \operatorname{sgn}\left(w_{i,2j+1}^{\mathfrak{B}}\right) & \leq & 0 \end{array} \tag{28}$$

This satisfies the biological property of the system to not posses any neuron, which can be simultaneously excitatory or inhibitory.

Given the initial states of the network $\mathfrak{B}$, the value of each neuron's state can be expressed as:

$$\begin{aligned} x_i^{\mathfrak{B}}(t) &= f\left(\sum_j^M w_{ij}^{\mathfrak{B}} x_j^{\mathfrak{B}}(t-1)\right) \\ &= f\left(\sum_j^N \left(w_{i(2j)}^{\mathfrak{B}} x_{2j}^{\mathfrak{B}}(t-1) + w_{i(2j+1)}^{\mathfrak{B}} x_{2j+1}^{\mathfrak{B}}(t-1)\right)\right) \end{aligned} \tag{29}$$

By definition when $t = 0$:

$$x_{2i}^{\mathfrak{B}}(0) = x_{2i+1}^{\mathfrak{B}}(0) = x_i^{\mathfrak{A}}(0) \tag{30}$$

If this holds for $t-1$, then equation 29 can be simplified as:

$$x_i^{\mathfrak{B}}(t) = f\left(\sum_j^N \left(w_{i(2j)}^{\mathfrak{B}} x_j^{\mathfrak{A}}(t-1) + w_{i(2j+1)}^{\mathfrak{B}} x_j^{\mathfrak{A}}(t-1)\right)\right) \tag{31}$$

For each pair $\left(w_{i2j}^{\mathfrak{B}}, w_{i2j+1}^{\mathfrak{B}}\right)$, by definition, either $w_{i2j}^{\mathfrak{B}} = w_{ij}^{\mathfrak{A}}$ and $w_{i2j+1}^{\mathfrak{B}} = 0$ (if $\operatorname{sgn}\left(w_{ij}^{\mathfrak{A}}\right) > 0$) or $w_{i2j}^{\mathfrak{B}} = 0$ and $w_{i2j+1}^{\mathfrak{B}} = w_{ij}^{\mathfrak{A}}$ (if $\operatorname{sgn}\left(w_{ij}^{\mathfrak{A}}\right) < 0$). On the other hand, $\forall i,j\colon w_{2ij}^{\mathfrak{B}} = w_{2i+1j}^{\mathfrak{B}}$. This further simplifies the equation to:

$$x_{2i}^{\mathfrak{B}}(t) = x_{2i+1}^{\mathfrak{B}}(t) = f\left(\sum_j^N w_{ij}^{\mathfrak{A}} x_j^{\mathfrak{A}}(t-1)\right) \tag{32}$$

Which proves that:

$$\left(\forall (i,j) \in \mathfrak{R}\colon x_i^{\mathfrak{A}}(0) = x_j^{\mathfrak{B}}(0)\right) \Rightarrow \left(\forall (i,j) \in \mathfrak{R}, \forall t \in \mathbb{N}\colon x_i^{\mathfrak{A}}(t) = x_j^{\mathfrak{B}}(t)\right) \tag{33}$$

□

A simple example is provided to illustrated this theorem. Given a network $\mathfrak{A}$ of size 2, with:

$$W^{\mathfrak{A}} = \begin{pmatrix} 11 & 12 \\ -21 & 22 \end{pmatrix} \tag{34}$$

The first neuron of this example has inhibitory and excitatory connections, the second neuron does not induce any problem. The weight matrix of the new network

$\mathfrak{B}$ is constructed as:

$$W^{\mathfrak{A}} = \begin{pmatrix} 11 & 12 \\ -21 & 22 \end{pmatrix} \qquad W^{\mathfrak{B}} = \begin{pmatrix} 11 & 0 & 12 & 0 \\ 11 & 0 & 12 & 0 \\ 0 & -21 & 22 & 0 \\ 0 & -21 & 22 & 0 \end{pmatrix} \tag{35}$$

It is obvious, from the matrix $W^{\mathfrak{B}}$, that the new network $\mathfrak{B}$ does not possess any neuron with both inhibitory and excitatory output. It can be observed that odd rows of the new matrix represent the excitatory influence, and the even rows represent the negative inhibitory influence of the corresponding original neuron. It is also obvious that for any given initial state $X^{\mathfrak{A}}$ both systems go through the same orbit for all corresponding neurons.

4. Performance Benchmark

This section compares the model presented in this chapter with a common implementation of a more biologically accurate model. To be more accurate this model needs to have a continuous time scale and a more complex activation function[11]. Which in turn implies a higher load on the computation of a simulation step.

The simulations are made for networks of growing size; for each network size, 100 simulations of 100 time steps are performed and the total time needed in second[12] is computed. Figure 3.4 on the following page shows the execution time in seconds for each model. The logarithmic scale of the plot (in Fig. 3.4 on the next page) shows the relative difference between the two models in a constant order of magnitude. This experiment clearly shows why simple neural network models are still be a strong alternative to more complex and accurate ones.

The continuous time simulation was done using a Runge–Kutta [see Butcher, 2003, for a review]. To be able to compare those two approaches, this section do as follows: Each use of the Runge–Kutta method generates the continuous evolution of the system over a given period of time. To help the comparison, the discrete time steps are considered to model the same period. In other words, each method of simulation is called the same number of times over the experience and thus the efficiency of each method is shown as a result of this test.

Being ten times faster is a good point for the model, but it needs to be powerful enough to solve complex problems specific to the cognitive brain models. The next chapters prove this, going from features of this model to a complete solution as a working memory.

5. Architecture

This section presents the global architecture of the system, which is used for the rest of this document. Figure 3.5 on page 75 shows the whole system. Each layer serves a specific purpose, detailed below, and is modeled by a certain number of neurons. When a specific region of the model is connected to another one, it is represented in this figure by an arrow and this arrow is labeled with a name (i.e. $\mathbf{W_R}$). Those labels are the names given to the corresponding sub-matrix of $\mathbf{W}$

[11]This function has to do integration step in order to produce the value of x.

[12]The benchmarks have been done on a modern day laptop in the same development framework with just a different activation function. See the annex for more details on the framework.

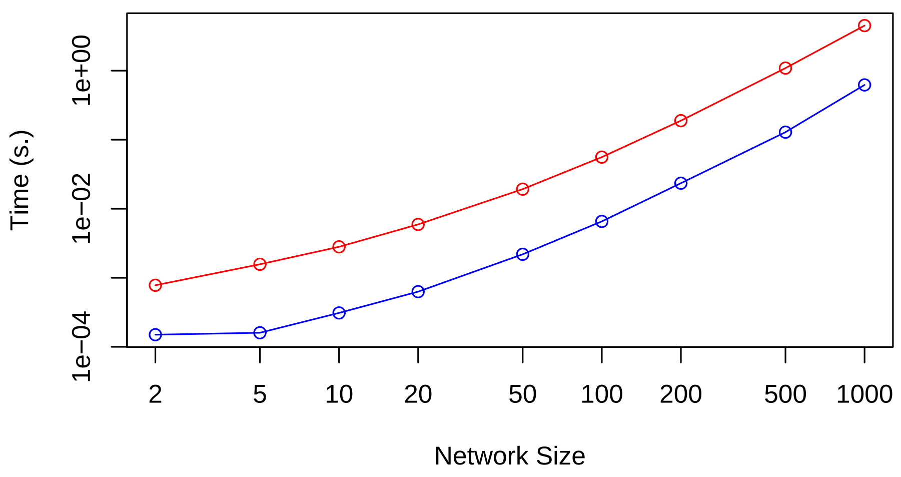


FIGURE 3.4. Simulation performance of two models. The benchmark compares the time taken by two models to perform 100 simulations of 100 time steps (the result is in second). As seen in this figure, the continuous time simulation (in red) is slower by one order of magnitude when compared to a discrete time simulation (in blue). However, the performance gap is independent of the size of the network.

which represents the corresponding weights. Here is an overview of the matrix $\mathbf{W}$ with all the sub-matrices:

$$\mathbf{W} = \begin{pmatrix} \mathbf{W_R} & \mathbf{W_V} & \mathbf{W_S} & \mathbf{W_I} \\ \mathbf{W_C} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{I} & 0 \\ \mathbf{W_A} & 0 & 0 & 0 \end{pmatrix} \tag{36}$$

The corresponding weights of a sub-matrix are referred to as w_{ij}^R, which is the connection weight from the jth neuron of A to the ith neuron in B if R links subsystem A to subsystem B. The state of each component can be seen in a vectorial form:

$$\mathbf{V} = \begin{pmatrix} \mathbf{X} & \mathbf{C} & \mathbf{S} & \boldsymbol{\Upsilon} \end{pmatrix} \tag{37}$$

At some places during the rest of this document, experiences are conducted using part of the model only, to highlight some specific features. This will be properly indicated by saying which part is not needed. To disable any part, the incoming and outgoing weights are set to zero (e.g: if a section does not work with the context, then $\mathbf{W_C} = \mathbf{W_V} = 0$); this also amounts to consider the corresponding subsystem as empty.

Those subsystems are referenced as:

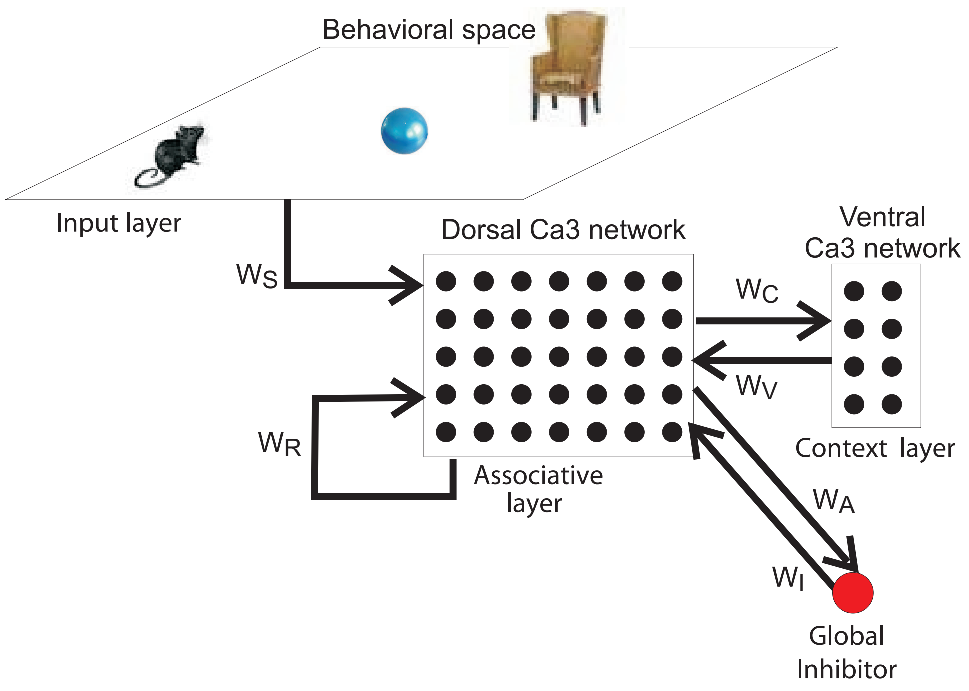


FIGURE 3.5. Architecture of the system, with all of its components as well as the connections that exist between them. If two components are connected, an arrow represents the direction and its label the corresponding sub-matrix. If two components do not see each other the arrow between them is just omitted, which means that the corresponding sub-matrix will be null.

- A: the associative layer, and $x_i(t)$ is the state of neuron $i \in A$ (noted $\mathbf{X}(t)$ in its vectorial form.)
- C: the context layer, and $c_i(t)$ is the state of neuron $i \in C$ (noted $\mathbf{C}(t)$ in its vectorial form.)
- Υ: the global inhibitor, and $\Upsilon(t)$ is the state of the inhibitor neuron.
- S: the input layer (aka the stimulus), and $\iota_i(t)$ is the state of neuron $i \in S$ (noted $\mathbf{S}(t)$ in its vectorial form.)

Since each part represents a different biological region, its modeling will try to reflect the properties of that region. To this end each group will present a different evolution function. During the presentation of each part, the activation function used on those neurons will be specified and explained.

5.1. Input layer: Behavioral Space. This layer represents the stimuli as they arrive to the system. For simplification purposes it is said to represent the external environment but this is not exactly accurate. Actually, the inputs, here, are pre-processed versions of the true external stimuli[13]. The input layer's neurons change but at a time scale much slower than other neurons presented in this model and not as a result of the dynamics of the system presented here, but due to external

[13]Hence, the usage of a mouse, ball, chair in its representation in Figure 3.5 (and in similar ones further in this thesis) should be considered as a direct representation of its contents. The dentate gyrus is a good candidate for this job in the brain.

circumstances[14]. So they will be considered as static values in this model, defined externally (usually by the user). Given this layer does not receive feedback from any other part of the system except itself with the identity matrix as weights (see equation 36 on page 74), if the activation function used for those neurons is set to be the identity function, the value of the input layer will never change. So overall an input is a fixed vector of size $|S|$, unless a perturbation is applied.

5.2. Associative layer: Dorsal Ca3 Network. This is the main layer, also referred to as the main module. It is the only recurrent part of the network, and the only one which is connected with all the other layers. In fact, all the layers are there to support and enhance the behavior of the associative layer. This layer serves as the main memory, and it is where information is stored. The recurrent connections also make it possible for this layer to exhibit very subtle outputs, which are highlighted, justified and praised in the next chapters. The threshold function used here is described in equation 22 on page 70.

5.3. Context layer: Ventral Ca3 Network. The context layer will be used in the last part of this document when the model is used as a working memory. It is used when the inputs are not simply considered as a vector of numbers but as a sequence of correlated entries which are related to a specific context (this is also called a "cognitive map"). In practice, a size for the cognitive maps is chosen [15] and then the inputs are generated by partitioning the input space into K sets of equal size (where K is the size of the cognitive map). Each set of input neurons represents an input of that cognitive map.

This introduces an extra level of information which is not present in the input itself, but is given to the system by the fact that a series of inputs has been encountered together in a short period of time.

Basically, the context layer acts as a classification network, which identifies for each stimulus its corresponding group. In order to reflect that, a "multiple winner takes all" activation function was chosen. Granted K_i, the total activity impinging (through $\mathbf{W_C}$) a context cell c_i, the activity of that cell at time t is given by the function $g_i\left(t\right)$, defined as:

$$g_i(t) = \begin{cases} 1 & \text{if } \left(\dfrac{K_i(t)}{\sum_i^P K_i(t)}\right)^{10} > 0.1 \\ 0 & \text{otherwise} \end{cases} \tag{38}$$

where $P = |C|$ is the total number of cells in the context layer. The K_i are the value of the vector $\mathbf{K}$ obtained as follow:

$$\mathbf{K} = \mathbf{W_C X} \tag{39}$$

The exponentiation enhances the distance between good candidates and averaged ones. This transfer function can also be written in a vectorial form as a function G:

$$G\colon \mathbb{R}^N \to \{0,1\}^N : \mathbf{X} \mapsto G\left(\mathbf{W_C X}\right) = \left[g_0 g_1 \dots g_N\right] \tag{40}$$

For the context layer to make sense, each stimulus must be associated to a given context. This information can be provided by a lot of ways to the system. In

[14]For instance because the experimenter wants to see the impact of some perturbation of the inputs on the rest of the system.

[15]Here, for convenience, the cognitive maps have been chosen to have a size of 5.

this model the information is an inherent part of the original stimulus. The goal of the context layer is twofold; first it is there to identify the correct context to which the current input belongs, secondly it gives the contextual information extracted from the input as a feedback to the associative layer (through $\mathbf{W_V}$).

This feedback mechanism allows this layer to help remove ambiguity. For example, if the system is faced with a partial input such as the word '**eel'. One can try to replace the missing letters by 'st' to form the word 'steel' or by 'wh' to form the word 'wheel'. It is obvious that without further information it is not possible to know which one is correct. However, if this noisy word is put in context in a sentence, such as "this blade is made of **eel" it becomes easy to choose the correct substitution. When the word 'blade' comes, it is recognized and enables a certain context (in the context layer) which in turn helps the associative layer when later the word '**eel' comes along.

In this model, since inputs are vectors in $[0,1]^N$ a possible modeling of this example can be, if $N = 40$, and if the inputs are given only values in $\{0,1\}$ then the input space can be seen as 40 bits of information where each 8 bits encode a letter. In this particular configuration, using the ascii encoding, the words 'wheel' and 'steel' become:

$$\begin{array}{lclllll} \text{wheel} & = & 01110111 & 01101000 & 01100101 & 01100101 & 01101100 \\ \text{steel} & = & 01110011 & 01110100 & 01100101 & 01100101 & 01101100 \end{array} \tag{41}$$

It is obvious from this example that if the two first blocs are missing in the input layer, the system is faced with a completely ambiguous information and cannot recover from it unless an external help is provided.

5.4. Global Inhibitor. The global inhibitor is there to prevent network saturation and to avoid a difficult tuning of the weight matrix: a competitive mechanism was implemented through global inhibition [see Rolls and Treves, 1998, for review]. To this aim, the global activity of the network was averaged by a single unit (called node 0), which in turn inhibits proportionally the entire network. Additionally, to avoid global stabilization of the network to the null fixed point, below a certain threshold of activity this unit excites the network instead. To produce such an effect, the activation function of this unit is redefined as :

$$\Upsilon(t) = \sigma \sum_{j=1}^{N} w_{0j}^{A} x_j(t-1) - \sigma N \beta \tag{42}$$

where β is the expected averaged activity of the network, σ is the strength of the global inhibition, and N the size of the associative layer A. The corresponding weights of this layer are defined as $w_{i0}^{I} = -1$ and $w_{0j}^{A} = 1$ (and $w_{i0}^{I} = w_{0j}^{A} = 0$ if the inhibitor is disabled). Accordingly, if the recurrent network's total activity is larger than $\sigma N \beta$, the inhibiting will give a negative feedback ($w_{i0}^{I} * \Upsilon(t) < 0$) to the associative layer a thus be inhibitory; however, when the activity is below that threshold, the unit output is becomes positive ($w_{i0}^{I} * \Upsilon(t) < 0$) and thus it becomes excitatory. Figure 3.6 on the following page show this activation function.

This function is also written in a vectorial form as the function H:

$$H\colon \mathbb{R}^N \to \mathbb{R}\colon \mathbf{X} \mapsto H(\mathbf{W_A X}) = \boldsymbol{\Upsilon}(t) \tag{43}$$

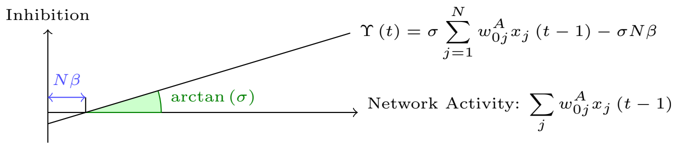


FIGURE 3.6. The inhibitor activation function. The x-axis shows the summed activity of the network impinging the inhibitor, and the y-axis shows the resulting inhibition. The higher the value, the stronger the inhibition. As shown on this plot, below a certain activity ($\sigma N\beta$) this unit becomes excitatory.

5.5. Simulation Rule. This part summarizes the simulation rule defined in equation 23 on page 71 with all the layers taken into account. The new definition is:

$$\begin{aligned}
\mathbf{X}(t) &= F\left(\mathbf{W_R X}(t-1) + \mathbf{W_S I}(t-1) + \mathbf{W_V C}(t-1) + \mathbf{W_I \Upsilon}(t-1)\right) \\
\mathbf{C}(t) &= G\left(\mathbf{W_C X}(t-1)\right) \\
\mathbf{S}(t) &= \mathbf{S}(t-1) \\
\mathbf{\Upsilon}(t) &= H\left(\mathbf{W_A X}\right)
\end{aligned} \tag{44}$$

6. Development of a neural toolbox

This thesis working on models that are extensively tested, and simulated on a computer, it needs a strong framework to support it. The work done here has never been straightforward, and often the aimed target is moving very fast; the only way to deal with this is to have a very flexible framework, which can help easily test new learning algorithms, neuronal architectures, and connectionist models. It was also useful from time to time to be able to reproduce results from the literature since there is always some small differences amongst models used by those people, and what is exposed here, and these small changes can lead to unexpected results. Finally, the system needs to be efficient, easy to use and extensible. To cope with all these requirements, a neural network framework was developed[16].

This framework was named **NSI**, for neural scripting interface, and is composed of two components: the neural development interface (NDI) and the neural development kit (NDK). The NDK is the core of this framework; it defines all the useful structures to help modeling all kinds of systems; it is a C++ library and it requires the knowledge of C++ programming to be used. The NDI is nothing more than an interface that supports custom scripts in the LUA language [Ierusalimschy et al., 2006], extended with special classes and functions defined in the NDK, in a friendlier user interface. The NDI makes it easy to test and see the results of all kinds of experiments (supporting plotting, parameter tuning, states/weights visualization,. . .), and it was really important and helpful. Yet its inner construction is not really relevant in the context of this work and is not detailed here[17]. On the other hand, it is useful to detail the development of the NDK and explain how all the concepts that are discussed through this thesis can be modeled in a modern programming approach. The interested reader can find a fully detailed explanation of this framework in the Appendix on page 187.

[16]To be more accurate, several versions of this framework have been produced during the past years.

[17]It is mostly C++ developments with libraries like QT, luabind, boost, . . .

7. Conclusion

This chapter has introduced the basis for the results presented in this thesis. The first part has covered existing related approaches and exposed the line of work taken here. It is clear that complex dynamics, memory-like systems and autonomous learning are central to lots of works and this thesis is no exception. The approach taken during this work was detailed, highlighting the common parts, but also the distinctions that exist with the state of the art.

An extensive description of the model used for all the experiments has been detailed and all of its components were detailed. The model proposed here is a very simple model based on the simplest possible definition of a unit. The simplicity of the system makes it very easy to adapt on a modern computer and has very good performances when manipulating the network. When compared to more subtle approaches, such as continuous time network, a huge performance gap can be observed. Yet, care must be taken with simplifications in order not to step away from biological evidence. At several places, it has been explained (or proved) if a described feature can be related to a biological fact and how it can be related to it.

The final step of this chapter was to propose a global architecture of the system as parts of it will be used during the rest of the work. This architecture is modular and each experiment explicitly mentions the set of modules that it requires. The goal of this architecture is to provide a unified model for all the work presented here in order to ease the transition from one set of experiments to another.

CHAPTER 4

Learning Algorithms

Synaptic plasticity is now widely accepted as a basic mechanism underlying learning and memory. There is experimental evidence that neuronal activity can affect synaptic strength, through both long-term potentiation and long-term depression [Bliss and Lomo, 1973]. Inspired by (or forecasting) this biological fact, a large number of learning "rules", specifying how activity and training experience change synaptic efficacies, has been proposed [Hebb, 1949; Sejnowski, 1977]; such learning rules have been essential for the construction of most models of associative memory [among others: Amari, 1977; Amari and Maginu, 1988; Amit, 1995; Amit and Mongillo, 2003; Brunel et al., 1997; Fusi, 2002; Hopfield, 1982]. In such models, the neural network maps the structure of information contained in the external and/or internal environment into embedded attractors. Since Amari [1972], Grossberg [1992] and Hopfield [1982] precursor works, the privileged regime to code information has been fixed point attractors. Many theoretical and experimental works have shown and discussed the limited storing capacity of these attractor networks[1] [see Amit and Fusi, 1994; Amit et al., 1987; Domany et al., 1995; Gardner, 1987; Gardner and Derrida, 1989, for review].

However, many neurophysiological reports tend to indicate that brain dynamics is much more complex than fixed points and is more faithfully characterized by cyclic and weak chaotic regimes [Babloyantz and Lourenço, 1994; Kenet et al., 2003; Nicolis and Tsuda, 1985; Rodriguez et al., 1999; Skarda and Freeman, 1987]. In line with these results, this chapter takes a closer look at what can be done with recurrent neural networks and complex dynamics. It is first showed why gradient descent algorithms are a bad candidate for memory-like applications. Later, a new algorithm to map stimuli to spatio-temporal limit cycle attractors of the network's dynamics is proposed. A learned stimulus is no more expected to stabilize the network into a steady state (which could in some cases correspond to a minimum of a Lyapunov function). Instead, the stimulus is expected to drive the network into a specific spatio-temporal cyclic trajectory. This cyclic trajectory is still considered as an attractor since content-addressability is expected. Before the presentation of the stimulus, the network could follow another trajectory and/or the stimulus could be corrupted with noise. As discussed in the earlier chapters, neural networks are dynamical systems and thus can exhibit complex behaviors; this chapter first shows when complex dynamics appear and why they are needed when recurrent neural networks are used as a memory-like structure. Later, it shows how they are a natural consequence of the Hebbian learning and to what extent the presence of complex dynamics can be used to improve the system even further.

By relying on spatio-temporal cyclic attractors instead of fixed-point attractors, no famous theoretical results, on the limited capacity of Hopfield networks, apply anymore. Actually, the extension of encoding attractors to cycles potentially boosts

[1]Attractor network is used for any model where the network's attractors are used as memory store.

this storing capacity. Suppose a network composed of two neurons that can only have two values: -1 and +1. Without paying attention to noise and generalization, only four fixed point attractors can be exploited whereas, by adding cycles, this number increases. For instance, cycles of length two are:

$$\begin{array}{lll} (+1,+1)(+1,-1) & (+1,+1)(-1,+1) & (+1,+1)(-1,-1) \\ (+1,-1)(-1,+1) & (+1,-1)(-1,-1) & (-1,+1)(-1,-1) \end{array}$$

This boost in the number of potential attractors justifies the use of cycles. However, indexing the memorized attractors restricted to the initial conditions, like classically done in Hopfield networks, would not allow a full exploitation of all these potential cyclic attractors, as it appears in the example above with multiple cycles sharing common patterns (e.g: depending on the cycle, the pattern $(+1,+1)$ gives $(+1,-1)$, $(-1,+1)$, or $(-1,-1)$). It makes their indexing based on parts of their content impossible without using an external information. This is the reason why in the experimental framework presented in this chapter, the indexing is rather done by means of the added external input layer, which continuously feeds the network with different external stimuli[2]. These external stimuli have another side effect: they modify the parameterization of the network, thus widen the possible dynamical regimes and the set of potential attractors.

The goal here is not to calculate the "maximum storage capacity" of these "cyclic attractors networks"[3], neither theoretically[4] nor experimentally[5]. Rather the goal is to show the potential behind the model described in Chapter 3 on page 73 and to discuss the potential dynamical regimes (oscillations and chaos) that allow this new form of information storing. Experimental results to be presented in the following chapter show how the theoretical limitations of fixed-point attractor networks can easily be overcome by the addition of the input layer together with the exploitation of the cyclic attractors.

The chapter starts out by showing the importance of complex dynamics for memories in the first section; these first results are done on simple random networks. They however indicate the need for a learning mechanism that can construct such a network. The second section describes different learning tasks and motivates two learning procedures. The "out-supervised" learning procedure is described in section three, where stimuli are encoded in predefined limit cycle attractors. The fourth section introduces the second learning algorithm, the "in-supervised" learning procedure, where stimuli are encoded in the limit cycle attractors which are derived from the ones spontaneously proposed by the network.

It is important to note that all the algorithms presented in this thesis are parametric and that their performances can be severely affected by those parameters. During the elaboration of those algorithms and their experimentation, different configurations have been tested empirically and the final used values are reported here. It is important to note that it is possible to go further and have a finer study of those parameters and their impact, and this can be the topic of further studies.

Before continuing further with this work, it is important to give a definition of 'the capacity'. Here, the capacity of the system is defined as the number of

[2]The external stimuli change by user input.

[3]To paraphrase Gardner [1987]'s paper: "The maximum storage capacity of neural networks".

[4]The fact that the work is not done at the thermodynamic limit (with infinite sized networks), as in population models, would render such an analyze very difficult.

[5]Some experimental limits do show up during the different tests but often it is difficult to know for sure if the algorithm has hit a capacitive limit or is just taking a very long time to learn the data set.

stable patterns that can be encoded and retrieved[6] in the network. This definition needs some subtlety when it comes to networks which learns cyclic patterns. Two measures are used: first the number of distinct stimuli which lead to a stable learned cyclic attractors[7], second the total number of patterns present in all the cyclic attractor learned in the system. Both those definitions can be used to compute the load of the system in terms of a ratio: the capacity of the system over the size of the system.

1. Complex dynamics

Molter and Bersini [2003a,b] reported the importance of complex dynamics for neural networks regarding their capacitive result, by showing the existence of a strong relation between the presence of complex dynamics in a system and its capacity. This section highlights the main points of this study. It is important to note that complexity of a system is a highly debated topic and many (not always compatible) visions exist. Here, the complexity of a system is always measured as its Lyapunov exponent or in other words it sensibility to initial condition.

The original idea takes its roots in the work of Hopfield [1982]. Few modifications were proposed to this approach, yet with the same idea behind the work: storing information in neural networks and studying their capacity. The first important modification is the absence of learning procedure. The reason for this was that the first goal in Molter's work was to show the general capacitive result in recurrent neural networks and to show that complex dynamics are a huge boost in this regards for those models. The second main difference between the two models is the presence of an external stimulus. Both models can be expressed in the general architecture proposed in the previous chapter and, while Molter uses the $\mathbf{W_S}$ weights and the external stimulus, Hopfield's model only works with $\mathbf{W_R}$. The impact of the stimulus can be easily understood by equation 44 on page 78 as modifying the inner dynamics of the model and thus providing different outputs for the same initial condition.

Molter noted, however, that the analysis of these systems was meaningless (in terms of memory) since the output of neurons were continuous and in that regards they theoretically provide an infinite storage capability with no noise tolerance. To work around that problem he suggested the use of a filter and to quantize the output of a neuron into r symbolic values as proposed by Omlin [2001]. This provides stronger output signals from the system since slight modifications in the output are lost after the filtering. The final step here was to guarantee the stability of a given attractor, in his work, Molter translates this as:

> Finally, in order to guarantee the robustness of the learning, in a way reminiscent of the original use of Hopfield networks as associative memory, the same dynamical output must be associated with a variety of external stimuli, all selected in a small hypersphere of diameter ϵ_{rob} around the original external stimulus to learn. Such an external stimulus $\bar{\iota}$ is said to be ϵ-robust iff $\forall \bar{\iota}' \,|\, |\bar{\iota}' - \bar{\iota}| \leq \epsilon$ we have the same quantized output. $|\bar{\iota}' - \bar{\iota}|$ is the euclidean distance between $\bar{\iota}'$ and $\bar{\iota}$.

[6]A pattern is learned if given a non noisy input the recovered pattern is a match with the one that was learned (e.g: Hamming distance of 0).

[7]A sequence of patterns.

This allowed him to study the encoding potential of the recurrent neural networks as the quantity of information which could be stored in ϵ-robust symbolic strings obtained from the quantization of the network's mean signal while being fed by different stimuli. The main reported result (shown in Figure 4.1) shows that in networks with huge potential, chaos prevails more and more as a natural and spontaneous dynamical regime, and stable systems show very poor capacitive results. This is a very interesting result since it is reminiscent of biological observations suggesting that chaos seems to be a crucial part of the brain [Skarda and Freeman, 1987].

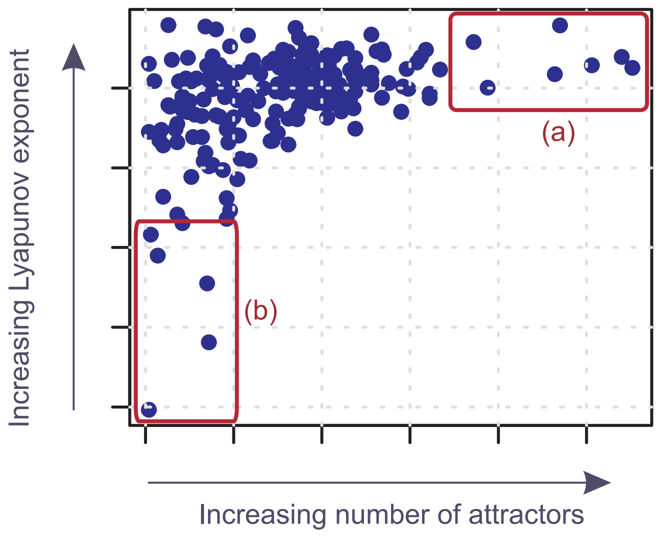


FIGURE 4.1. Capacitive vs. complexity. Each point represents one random network, the complexity is measured as an mean maximum Lyapunov exponent, and the capacity is computed as the number of ϵ-robust (stable) attractors. From this plot, it appears that (a) huge capacity implies a network with strong complex dynamics and (b) stable networks have a poor capacity.

Two roles are suggested for the chaos appearing in these neural networks. First, chaotic activity enables the rapid state transitions, essential for information processing. This is crucial since it allows the system to move through different region of the system without external constraint and thus permits to the system to rapidly provide responses to a given stimulus. Secondly, from a cognitive point of view, chaos is essential for the creation of information, as suggested by authors like Kentridge [2000]; Skarda and Freeman [1990b]; Tsuda [2001]: chaotic behavior is necessitated by the brain in order to perform efficient non-deterministic symbolic computations during cognitive tasks.

2. The Learning Task

From the evidence that chaos is a necessary condition for high capacity memory (see Figure 4.1), the next logical step was to work out a learning algorithm that can achieve such network configurations. This algorithm must have two important properties. First, it must promote complex behaviors for the system, to avoid it from becoming stable, which has been shown to be of very limited use as a memory. Secondly, the learning algorithm should not rely on the prediction of the trajectory, such as the gradient descent [Rumelhart et al., 1986a,b; Williams and Zipser, 1990], since complex systems will have lots of bifurcations which in turn make it impossible to rely on an information such as the slope of the trajectory [Molter, Salihoglu, and Bersini, 2004a].

2.1. Back-Propagation Through Time Algorithm. This part takes a look at the classical gradient descent algorithm called "back-propagation through time"(BPTT) and shows how this type of supervised learning fails to provide any convincing result when used for hetero-associative learning. This algorithm is based on the idea of error propagation of the classical back-propagation algorithm [Werbos, 1974]. However, this cannot be applied as is since the network is recurrent and thus does not possess any specific input/output neuron to propagate the information to/from. The idea behind this algorithm was to create a new network from the recurrent network, which will have two features: it will be a feed forward network and each layer will represent the original network at successive time steps. This is easily obtained by a process called "unfolding", which consists in stacking identical copies of the original network and redirecting connections from one copy to the following one in a reminiscent way of the original connections[8], the last copy has no connection. The number of copies is the maximum number of time steps after which the network is expected to converge. This process is represented in Figure 4.2.

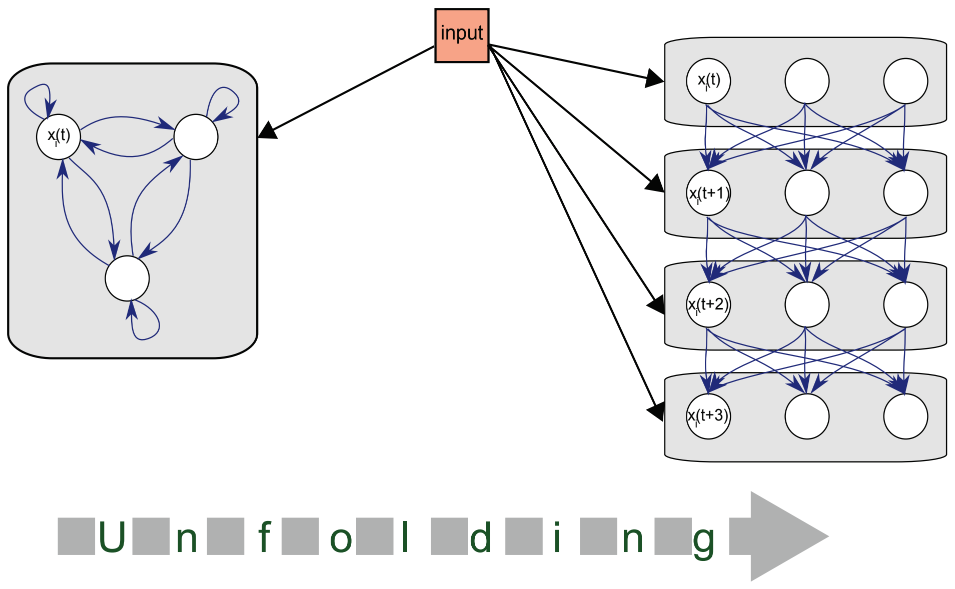


FIGURE 4.2. Unfolding process of a 3-neuron recurrent network, over 4 time steps.

The BPTT is defined over a data set composed of multiple pairs (an external stimulus and a symbolic output string), and follows these steps:

(1) **Forward pass**: all networks are updated from the first copy to the last one.

[8]For example, if a connection exists between neuron 1 and 2, the neuron 1 from copy one will be redirected to the neuron 2 of the second copy.

(2) **Error propagation**: the term $\delta_i(t)$ is computed for each time t and neuron i. This error is propagated backward through time[9], from $t = T, \cdots, 1$, with $\delta_i(T+1) = 0$ for convenience.

$$\begin{aligned} \delta_i(t) &= f'_{it}\left[\sum_k^N \delta_k(t+1)\, w_{ki} + (o_i(t) - x_i(t))\right] && \text{if } i = \text{output unit} \\ \delta_i(t) &= f'_{it}\left[\sum_k^N \delta_k(t+1)\, w_{ki}\right] && \text{if } i \neq \text{output unit} \end{aligned} \tag{45}$$

where o_{it} is the expected output for neuron i at time t, and N the size of the network.

(3) **Weight adjustement**: the weights of the recurrent connections from each neuron j are adjusted according to:

$$\Delta w_{ij} = \varepsilon \sum_{t=1}^{T} \delta_i(t)\, x_j(t-1) \tag{46}$$

and the weights from each input ι are adjusted according to:

$$\Delta w_{i\iota} = \varepsilon \sum_{t=1}^{T} \delta_i(t)\, o_\iota(t) \tag{47}$$

where ε is the learning rate.

After each pass, the network is simulated a few steps in order to avoid transient effects (around 10 time steps). After the training, the global error is computed. The algorithm stops once this error is lower than some specified threshold. If this error is bigger than or equal to the error computed at the previous time step, the weight connections are randomly reinitialized from a uniform distribution.

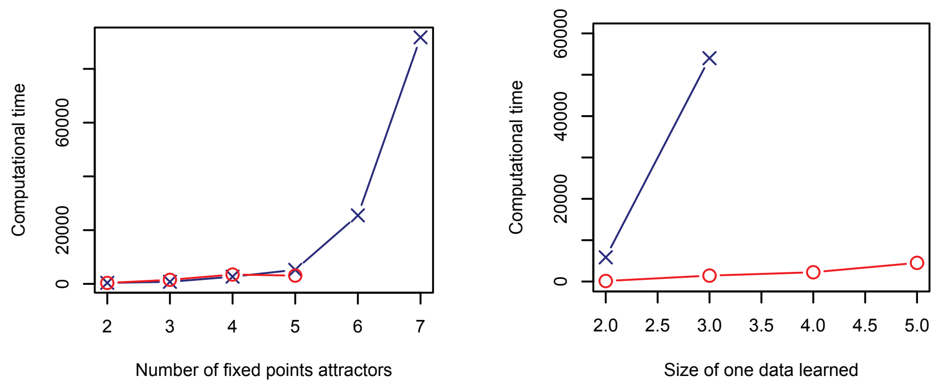


FIGURE 4.3. Comparison of learning time: BTTP vs random brute force. The BPTT learning algorithm (blue crosses) time to learn a given data set compared to the time it takes to find a random network (red circles) with that much attractors in it. On the left, are shown the results for encoding stimuli in fixed-point attractors. On the right, are shown the results for encoding one stimulus in a limit cycle of variable size. It is important to note that random networks fail to find any results for more than 5 fixed points attractor, while BPTT fails to learn a cyclic attractor with a period greater than 3. The networks have 4 neurons.

[9]Hence the name of the algorithm.

Here, the learning task consists in the hetero-association between external stimuli and quantized symbolic strings obtained from the mean signal of the network. The learning process can be improved by adding noisy versions of the data present in the data set. Following this training process, two important results are found. The first finding is on the computational performance of the BPPT algorithm. When comparing it to the raw potential of random networks [Molter and Bersini, 2003a,b], it seems that the BPTT is not a very good candidate for such applications. It struggles to learn data sets of several fixed point attractors and performs even worse for learning a single cyclic data of any reasonable length (see Figure 4.3 on the facing page). The BPTT's performances were compared with a brute force algorithm, which generates random networks until one contains the correct output. The Figure 4.3 on the preceding page clearly shows that the BPTT is, at best, in the same order of magnitude as the random process, which is not a very good sign.

However, several important details should be noted. When the number of fixed points to learn arises above 5, the execution time of the BPTT brutally increases, but the random procedure is still worse, by far. On the contrary, when learning cyclic attractors of increasing size, the random procedure performs well while the BPTT rapidly becomes unfeasible. The BPTT's performance can be drastically increased by fine parameter tuning according to the problem, but this is a non-trivial task.

Where the BPTT really shines and clearly outperforms a brute force search is when the spatial configuration of attractors is specified. In the test performed (and shown) in Figure 4.3 on the facing page, each individual neuron is free to have any output as long as the average of the system describes the desired output. In other words, when the system is expected to go through a specific output for each of its neurons it becomes extremely hard to find a random configuration to fit the requirements. The first conclusion that can be drawn for the BPTT is that it is more suited for learning static data.

The learning algorithm only specifies how the network is supposed to behave for learned stimuli; here the behavior of the system is tested when facing previously unlearned stimuli. The goal of the next experiments is to see what kind of global dynamics exists in the network because of the learning algorithm, and seeks the state space to look how many other stable attractors can be found. Figure 4.4 on the following page repeats the experiments performed in the previous section (see Figure 4.1 on page 84) but this time the networks are not random but taken after learning a three fix-point data set with both algorithms (BPTT and brute force). As it can be seen from these results the BPTT needs very stable networks to converge, which in turn gives a network with poor background dynamics, and this means poor capacity. The reason BPTT needs a very stable network to converge can be understood by looking at the Figure 4.5 on the following page, which shows how bifurcation can get in the way of a gradient descent. As seen in this figure a small modification in the weights to minimize the error will lead to a sudden jump due to the bifurcation.

These results show why gradient descent algorithms are not very good candidates for this line of work.

2.2. Iterative supervised Hebbian algorithm. The key difference between the gradient-based algorithms defined in the previous section and Hebbian algorithms comes down to locality. While the previous algorithm works at the scale of the network, Hebbian learning relies only on information available locally and thus adjusts each weight w_{ij} in function of the values of its direct neighbors (in

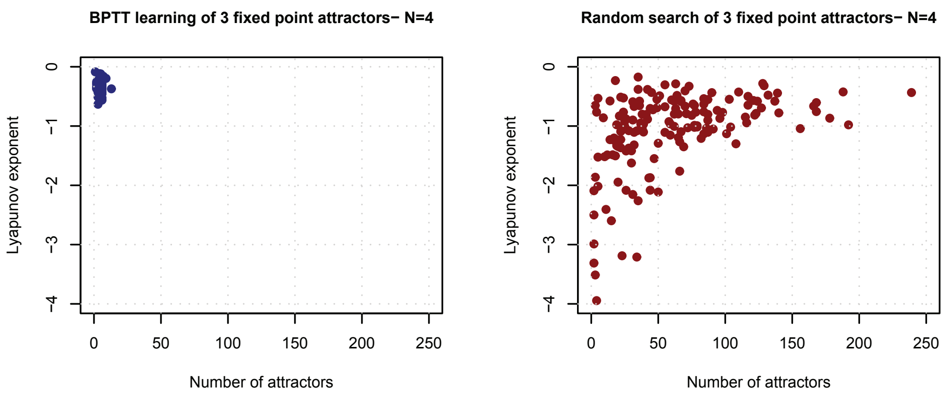


Figure 4.4. Capacity vs. complexity after learning. The experiments from the previous section (see Figure 4.1 on page 84) is repeated. However, this time, the networks are not generated randomly. The networks from the left figure are obtained after learning to associate 3 stimuli to distinct fixed points attractors with BPTT. The figure on the right shows networks that did the same learning task but with the random brute force approach. The number of potential attractors is very low for the BPTT, while some random networks show very good performances (up to 250 stable attractors). The networks have 4 neurons.

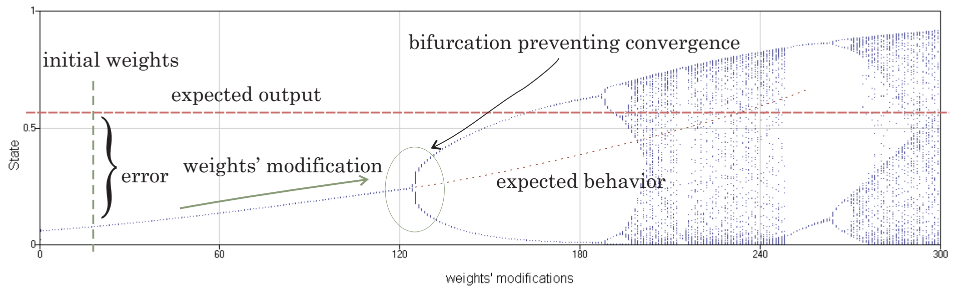


Figure 4.5. Bifurcation prevents gradient descent. Example of how a bifurcation can prevent a gradient descent algorithm to converge.

other words, the two neurons i and j it connects). Another important distinction of the Hebbian algorithm is the type of patterns it learns. The previous examples[10] learned the mapping of a specific stimulus to a symbolic string observed on the mean signal of the system (after quantization). Hebbian learning maps the stimuli into an attractors' spatio-temporal configuration. In other words, given a stimulus, the system learns to converge to a specific attractor specified in space (with the value of each neuron) and time (with the value of the neurons at different successive time steps).

In the next two sections, two new supervised Hebbian learning algorithms are proposed [Molter, Salihoglu, and Bersini, 2007b]. This section describes their common learning task. In fact the main philosophical difference between the two algorithms is the way the output for a given stimulus is specified (a priori predefined

[10]Using the BPTT.

vs. internally generated). Both consist in storing a set of q external stimuli in spatio-temporal cycles of the network's internal dynamics. The iterative Hebbian algorithm is not new, but the inclusion of n input layers is, and the use of cyclic patterns has never been studied to any depth. The first algorithm proposed in the next sections is an iterative Hebbian learning algorithm with those two additions, whereas the second algorithm is completely original and built on top of the first one. The data set is written:

$$\mathcal{D} = \left\{\mathcal{D}^1, \cdots, \mathcal{D}^q\right\} \tag{48}$$

The number of data stored in the network is quantified by a load parameter $\alpha = q/N$, where $N = |X|$ is the size of the associative layer. Each data $\mathcal{D}^\mu$ ($\mu \in \{1, \ldots, q\}$) is defined by a pair composed of:

- a pattern χ^μ corresponding to the external stimulus feeding the network;
- a sequence of patterns $\varsigma^{\mu,i}, i = 1, \cdots, l_\mu$ to store in a dynamical attractor.

$$\mathcal{D}^\mu = \left(\chi^\mu\,,\,(\varsigma^{\mu,1}, \cdots, \varsigma^{\mu,l_\mu})\right) \qquad \mu = 1, \cdots, q \tag{49}$$

where l_μ is the period of the sequence ς^μ and may vary from one data to another. Each pattern μ is defined by assigning digital values to all neurons:

$$\begin{array}{lclll} \chi^\mu & = & \{\chi_i^\mu,\ i = 1, \cdots, M\} & \texttt{with}\ \ \chi_i^\mu & \in \{-1, 1\} \\ \varsigma^{\mu,k} & = & \{\varsigma_i^{\mu,k},\ i = 1, \cdots, N\} & \texttt{with}\ \ \varsigma_i^{\mu,k} & \in \{-1, 1\} \end{array} \tag{50}$$

Where $N = |X|$ is the size of the associative layer and $M = |S|$ is the size of the input layer. The sequence of patterns or cycle size are specific to each experiment and are specified properly before hand, in the next Chapter that presents those experiments.

By suppressing the external stimulus and by defining all the sequences' periods l_μ to 1, this task is reduced to the classical learning task originally proposed by Hopfield: the storing of patterns in fixed-point attractors of the underlying RNN's dynamics. The learning task described above turns out to generalize the one proposed by Hopfield. To ease the reading, when patterns are stored in fixed-point attractors, they are noted: ξ^μ.

Before testing the learning of sequences, the algorithm has first to be validated by storing sets of static patterns in RNNs, without external stimuli, as in the original Hopfield model (see next Chapter). Thus, the data set becomes:

$$\mathcal{D}^\mu = \xi^\mu \qquad \mu = 1, \cdots, q \tag{51}$$

In the second test, a data becomes a pair composed of the external stimulus and the stored pattern. To validate the recourse to the external stimuli, and to enable a comparison with precedent results, the first tests use external stimuli as duplicated information:

$$\mathcal{D}^\mu = (\xi^\mu, \xi^\mu) \qquad \mu = 1, \cdots, q \tag{52}$$

then, the storing of hetero-associative data is tested:

$$\mathcal{D}^\mu = (\chi^\mu, \xi^\mu) \qquad \mu = 1, \cdots, q \tag{53}$$

and finally the coding of external stimuli in limit cycle attractors are tested.

3. The "Out-supervised" Learning Algorithm

This first learning task is very straightforward and consists in storing a well-defined data set. It means that each data stored in the network is fully specified a priori: each external stimulus must be associated to a pre-specified limit cycle attractor of the network's dynamics.

3.1. Introduction: Hopfield's auto associative model. In the basic Hopfield [1982] model, all connections need to be symmetric, no auto-connection can exist and the update rule must be synchronous. Hopfield has proven that these constraints are sufficient to define a Lyapunov function H for the system[11]:

$$H = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij} x_i \, x_j \tag{54}$$

Each state variation produced by the system's equation entails a non-positive variation of H: $\Delta H \leq 0$. The existence of such a decreasing function ensures a convergence to fixed-point attractors. Each local minimum of the Lyapunov function represents one fixed point of the dynamics. These local minima can be used to store patterns. This kind of network is akin to a content-addressable memory since any stored item will be retrieved when the network dynamics is initiated with a vector of activation values sufficiently overlapping the stored pattern[12]. In such a case, the network dynamics is initiated in the desired item's basin of attraction, spontaneously driving the network dynamics to converge to this specific item.

The set of patterns can be stored in the network by using the following Hebbian learning rule, which obviously respects the constraints of the Hopfield model (symmetric connections and no auto-connection), given a data set of p elements:

$$w_{ij} = \frac{1}{N}\sum_{\mu=1}^{p} \xi_i^\mu \, \xi_j^\mu \qquad w_{ii} = 0 \tag{55}$$

Where ξ_i^μ is the i^{th} element of the μ^{th} pattern of the data set.

However, this kind of rule leads to drastic storage limitations. An in-depth analysis of the Hopfield model's storing capacity has been done by Amit et al. [1987] by relying on a mean-field approach and on replica methods originally developed for spin-glass models. Their theoretical results show that such a type of networks, when coupled with this learning rule, is unlikely to store more than $0.14\,N$ uncorrelated random patterns.

3.2. The iterative version of the Hebbian learning rule.

3.2.1. *Learning fixed-points.* A better way of storing patterns is given by an iterative version of the Hebbian rule [Gardner, 1987], [see Forrest and Wallace, 1995; van Hemmen and Kuhn, 1995, for a detailed description of this algorithm]. The principle of this algorithm is as follows: at each learning iteration, the stability of every nominal pattern ξ^μ, is tested. Whenever one pattern has not reached stability yet, the responsible neuron i sees its connectivity reinforced by adding a Hebbian term to all the synaptic connections impinging on it:

$$w_{ij} \mapsto w_{ij} + \varepsilon_s \, \xi_i^\mu \, \xi_j^\mu \tag{56}$$

[11]A Lyapunov function defines a positive lower bounded monotonically decreasing function.

[12]This remains valid only when learning a few number of uncorrelated patterns. In other cases, the network may converge to any fixed-point attractor. Here, the correlation of two patterns is measured as the number of elements they share.

where ε_s defines the learning rate. All patterns to be learned are repeatedly tested for stability and, once they are all stable, the learning is complete. To test the stability of a fixed-point pattern, the algorithm has to check if the pattern is preserved after each time step. This is done by setting the states of the system so that they reflect a given pattern, then simulate the system for one time step and if the Hamming distance between the new obtained pattern and the original one is 0, then the system is stable.

This learning algorithm is incremental since the learning of new information can be done by preserving all information that has already been learned. It has been proved [Gardner, 1987] that by using this procedure, the capacity can be increased up to $2N$ uncorrelated random patterns.

In the model presented in this thesis, stored cycles are indexed by the use of external stimuli. These external stimuli are responsible for a modification of the underlying network's internal dynamics and, consequently, for increasing the number of potential attractors, as well as the size of their basins of attraction. The connection weights between the external stimuli and the neurons are learned by adopting the same approach as given in Equation 56 on the preceding page. When one pattern is not stable yet, the responsible neuron i sees its connectivity reinforced by adding a Hebbian term to all the synaptic connections impinging on it (Eq. 56 on the facing page), including connections coming from the external stimulus:

$$w_{ik} \mapsto w_{ik} + \varepsilon_b \, \chi_k^\mu \, \xi_i^\mu \tag{57}$$

where ε_b defines the learning rate applied on the external stimulus' connections, which may differ from ε_s.

In order to not only store the patterns, but also to ensure a sufficient content-addressability, the learning procedure needs to try to "excavate" the basins of attraction. Two approaches are commonly proposed in the literature. The first approach aims at getting the alignment of the spin of the neuron ($+1$ or -1) together with its local field to be not just positive (which is the requirement to ensure stability), but greater than a given minimum bound. The second approach attempts at explicitly enlarging the domains of attraction around each nominal pattern. To do so, the network is trained to associate noisy versions of each nominal pattern with the desired pattern, following a given number of iterations expected to be sufficient for convergence. This last approach is the one adopted here. It was first suggested by Forrest and Wallace [1995] that, when learning fixed-point attractors, when α[13] gets close to 0.5, the basins of attraction become negligibly small without such a mechanism put into place.

Because of these two observations, two noise parameters are introduced to tune noise during the learning phase: the noise imposed on the internal states ln_s and the noise imposed on the external stimulus ln_b[14]. The noise parameter represent the percentage of a pattern that should be affected by noise. To apply this noise a uniform random sampling without re-pick is done until enough units are selected. Then those unit are shifted by multiplying their value by -1.

A noisy pattern $\xi^\mu_{ln_s}$ is obtained from a pattern to learn ξ^μ by choosing a set of $|A| \, ln_s$ items, randomly chosen among all the initial pattern's items, and by switching their sign. Thus, $d_H(ln_s)$ defines the Hamming distance[15] between the

[13] α is the load parameter of the network as defined in section 2.2 on page 87.

[14] There is no real way to determine a good value for these parameters and their value usually depends on the problem.

[15] All Hamming distance are normalized to range in $[0, 100]$ for sake of easier comparison

two patterns:
(58)

$$d_H(ln_s) = \sum_{i=1}^{N} d_i \quad \texttt{where} \left\{ \begin{array}{llll} d_i = 0 & \texttt{if} & \xi_i^\mu \, \xi_{i,ln_s}^\mu = 1 & \texttt{(equal items)} \\ d_i = 1 & \texttt{if} & \xi_i^\mu \, \xi_{i,ln_s}^\mu = -1 & \texttt{(different items)} \end{array} \right.$$

And $d_H(ln_b)$ defines the Hamming distance[16] between the two stimuli:
(59)

$$d_H(ln_b) = \sum_{i=1}^{N} d_i \quad \texttt{where} \left\{ \begin{array}{llll} d_i = 0 & \texttt{if} & \chi_i^\mu \, \chi_{i,ln_s}^\mu = 1 & \texttt{(equal items)} \\ d_i = 1 & \texttt{if} & \chi_i^\mu \, \chi_{i,ln_s}^\mu = -1 & \texttt{(different items)} \end{array} \right.$$

3.2.2. *Learning cycles.* The learning rule (defined in Equation 56 on page 90) naturally leads to asymmetrical weights' values. It is no longer possible to define a Lyapunov function for this system, the main consequence being to include cycles in the set of "memory bags".

As for fixed-points, the network can be trained to converge to such limit cycle attractors by modifying the equations (56 and 57 on the previous page). This time, the weights w_{ij} and w_{is} are modified according to the expected value of neuron i at time $t+1$ and the expected value of neuron j and of the external stimulus at time t (see Equation 61 in Algorithm 1 on page 98):

$$\begin{array}{lcl} w_{ij} & \mapsto & w_{ij} + \varepsilon_s \, \varsigma_i^{\mu,\nu+1} \, \varsigma_j^{\mu,\nu} \\ w_{is} & \mapsto & w_{is} + \varepsilon_b \, \varsigma_i^{\mu,\nu+1} \, \chi_i^{\mu} \end{array} \tag{60}$$

where ε_s and ε_b respectively define the learning rate and the stimulus learning rate[17]. This time asymmetric Hebbian rules can be related with asymmetric time windows of synaptic plasticity observed in pyramidal cells during tetanic stimulus conditions [Bi and Poo, 1999; Levy and Steward, 1983].

Algorithm 1 on page 98 describes the pseudo-code used to store cycles (storing static patterns or hetero-associative memories appear as a limit case of this algorithm). The two enhancements described above are included: noisy patterns are added during the training period to increase robustness to noise and external stimuli are used to enhance both the storing capacity and the robustness again.

Equations 62 in Algorithm 1 on page 98 are the central equations responsible for the robustness to noise. It works as follows: after initializing the network with noisy data and skipping the transient period of the orbit, the network stabilizes to an attractor. A comparison is done with the desired attractor - if they are not equal, weights are modified according to the Hebbian rule driving each step of the cycle to give the following one in the next iteration. The robustness is guaranteed provided the noisy version of any member of the cycle is associated with the non noisy version of the following one.

3.2.3. *Adaptation of the algorithm to continuous activation functions.* When working with continuous state neurons, the system is no more working with cyclic attractors but with limit cycle attractors. Therefore, the algorithm needs to be adapted in order to prevent the learned data from vanishing after a few iterations. One-step iteration does not guarantee long-term stability of the internal states since observations are performed using a filter layer. The adaptation consists in waiting

[16] One must note that the Hamming distance applied to patterns valued in $\{0,1\}$ is nothing more than the 1-norm Euclidean distance: $\sum_{i=0}^{n} |x_i - y_i|$.

[17] Those parameters are usually chosen small to avoid sudden changes. However, choosing them too small reduces the performance of the learning algorithm. They are usually chosen around 0.05.

a certain number of cycles before testing the correctness of the obtained attractor. The halting test for discrete neurons is given by the following equation:

$$\forall \mu, \nu \texttt{ if } (\mathbf{x}(0) = \varsigma^{\mu,\nu}) \mapsto \left(\mathbf{x}(1) = \varsigma^{\mu,\nu+1}\right) \tag{63}$$

while, for continuous neurons, it becomes:

$$\forall \mu, \nu \texttt{ if } (\mathbf{x}(0) = \varsigma^{\mu,\nu}) \mapsto \left(\mathbf{o}(1) = \mathbf{o}(l_\mu + 1) = \cdots = \mathbf{o}(T * l_\mu + 1) = \varsigma^{\mu,\nu+1}\right) \tag{64}$$

where T is a further parameter of the algorithm (set to 10 in all the experiments) and $\mathbf{o}$ is the filtered output vector defined in Equation 20 on page 67.

The easiest way to understand the need for the second halting test when dealing with continuous state networks is to consider the simplest case of a one-neuron network which has to learn one fixed point attractor $\mathcal{D}^0 = \left(\chi^0, \xi^0 = 1\right)$. When dealing with a discrete state network, if $\left(x\left(0\right) = \xi^0\right) \mapsto \left(x\left(1\right) = \xi^0\right)$ then it is clear the system has converged since it is deterministic. However, even this simple example becomes tricky with continuous state networks. For example, given the following orbit:

$$(x(0) = 0.5) \mapsto (x(1) = 0.3) \mapsto (x(2) = 0.1) \mapsto (x(3) = -0.1) \mapsto (x(4) = -0.2) \tag{65}$$

This orbit passes the first halting test since when looked through the filter layer it gives:

$$(x(0) = 1) \mapsto (x(1) = 1) \mapsto (x(2) = 1) \mapsto (x(3) = -1) \mapsto (x(4) = -1) \tag{66}$$

Which in turn satisfies $\mathbf{o}(0) = \mathbf{o}(1)$. However, it its clear that this orbit is not valid and does not seem to stabilize at 1. Using the second halting test with a carefully[18] chosen value for T (here $T \geq 3$), the test fails, since $\mathbf{o}(0) = \mathbf{o}(1) = \mathbf{o}(2) \neq \mathbf{o}(3) = \mathbf{o}(4)$. The Figure 4.6 on the following page shows how a learned data vanishes when the first halting test is used on a continuous states network.

4. The "In-supervised" Learning Algorithm

As shown in Section 1 on page 101, the encoding capacities of networks learning in the "out-supervised" way described above are good. However, these results are disappointing compared to the potential capacity observed in random networks [Molter and Bersini, 2003a,b]. Moreover, section 2 on page 110 shows how learning too many cycle attractors in an "out-supervised" way leads to the kind of blackout catastrophe similar to the ones observed in fixed point attractors networks [Amit, 1989]. Here, the network's background regime becomes fully chaotic and similar to white noise.

Learning pre-specified data appears to be too constraining for the network. This section introduces an "in-supervised" learning algorithm, which removes external supervision: the network has to learn to react to an external stimulus by cycling through a sequence which is not specified a priori but which is obtained following an internal mechanism. In other words, the information is "generated" through the learning procedure that assigns a "meaning" to each external stimulus. There is an important tradition of "less supervised" learning in neural networks since the seminal works of Kohonen [1998] and Carpenter and Grossberg [1988]. This tradition enters in resonance with writings in cognitive psychology and constructivist philosophy [Erdi, 1996; Piaget, 1963; Tsuda, 2001; Varela et al., 1991,

[18]Again, finding a good value for this parameter requires trial–and–error and depends on the problem.

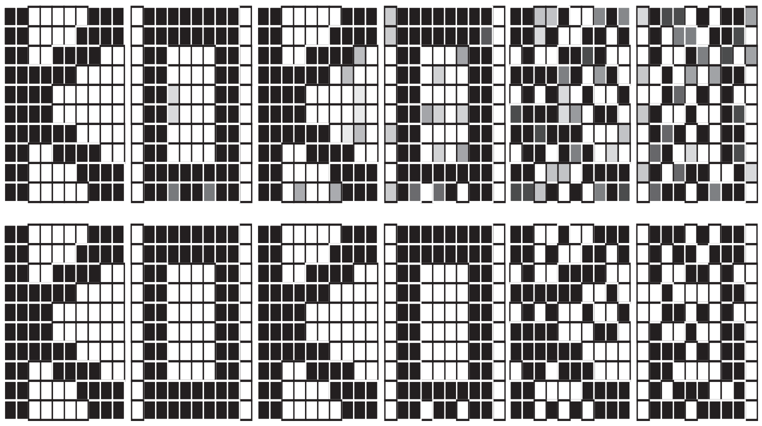


FIGURE 4.6. Stability of cycles in continuous networks. Here a network of 100 neurons is arranged in a grid of 10×10. Each column is one time step of simulation. The upper part is the inner state of the neuron plotted in a gray scale where the bottom part shows the filtered output. The cycle "KO" has been learned by the network with the parameter T set to 0. After the initialization to one letter of the sequence (the letter "K"), the network cycles through the entire learned sequence ("KO"). However, the internal state diverges slightly, but this alteration is not visible on the filtered layer, which may give the impression of convergence. However, at the end of the second cycle, differences appear in the output. The learned sequence vanishes progressively. This shows the importance of the parameter T.

among others]. The algorithm to be presented now can be seen as a dynamical extension of the "out-supervised" algorithm, in the spirit of these preliminary works, where the coding scheme relies on cycles instead of single neurons.

4.1. Description of the learning task. The main characteristic of this new algorithm lies in the nature of the learned information: only the external stimuli are known before learning. The limit cycle attractor associated with an external stimulus is identified through the learning procedure: the learning procedure enforces a mapping between each stimulus of the data set and a limit cycle attractor of the network's inner dynamics, whatever it is. Hence, the aim of the learning procedure is twofold: first, it proposes a dynamical way to code the information (i.e. to associate a "meaning" to the external stimuli), then it learns it (through a classical supervised procedure).

Before mapping, the data set is defined by: $\mathcal{D}_{bm}$ (bm standing for "before mapping"):

$$\mathcal{D}_{bm} = \left\{\mathcal{D}_{bm}^1, \cdots, \mathcal{D}_{bm}^q\right\} \qquad \mathcal{D}_{bm}^\mu = \chi^\mu \qquad \mu = 1, \cdots, q \tag{67}$$

After mapping, the data set becomes:

$$\mathcal{D}_{am} = \left\{\mathcal{D}_{am}^1, \cdots, \mathcal{D}_{am}^q\right\} \qquad \mathcal{D}_{am}^\mu = \left(\chi^\mu, (\varsigma^{\mu,1}, \cdots, \varsigma^{\mu,l_\mu})\right) \qquad \mu = 1, \cdots, q \tag{68}$$

where l_μ is the period of the learned cycle.

4.2. Description of the algorithm. The main difference between this algorithm and the one introduced in the previous section lies in the nature of the information learned. In the "out-supervised" version, each information to learn is given a priori and is fully specified. In the "in-supervised" version, only the external stimuli are given a priori: the information is a consequence of the learning. In other words, the information is "generated" through the learning procedure assigning a "meaning" to each external stimulus: the learning procedure enforces a mapping between each stimulus of the data set and a limit cycle attractor of the network's inner dynamics, whatever it is.

Inputs of this algorithm are:

- a data set $\mathcal{D}_{bm}$ to store (Equation 67 on the preceding page);
- a range $[\texttt{min}_{cs}, \texttt{max}_{cs}]$ which defines the bounds[19] of the accepted periods of the limit cycle attractors coding the information;

This algorithm can be broken down in three phases, which are constantly iterated until convergence:

re-mapping stimuli into cyclic attractors: During this phase, the network is presented with an external stimulus which drives it into a temporal attractor $\mathbf{output}^{\mu}$ (which can be chaotic). Since the idea is to constrain the network as little as possible, a meaning is assigned to the stimulus by associating it with a close cyclic version of the attractor $\mathbf{output}^{\mu}$: called $\mathbf{cycle}^{\mu}$, it is an original[20] attractor respecting the periodic bounds $[\texttt{min}_{cs}, \texttt{max}_{cs}]$. This step is iterated for all the stimuli of the data set;

learning the information: Once a new attractor $\mathbf{cycle}^{\mu}$ has been proposed for each stimulus, they are tentatively learned by means of a supervised procedure. However, to avoid constraining the network too much, only a limited number of iterations are performed, even if no convergence has been reached. For details see Algorithm 2 on page 99;

end test: if all stimuli are successfully associated with different cyclic attractors, the "in-supervised" learning stops, otherwise the whole process is repeated.

See Algorithm 2 on page 99 for the pseudo-code of the "in-supervised" learning algorithm.

It has to be noted that this complete learning mechanism implicitly supplies the network with an important robustness to noise. First, the coding attractors are the ones naturally proposed by the network. Attractors that are a priori specified are (in most) case very far from the dynamics that the system outputs for that stimulus, thus they require heavy weight modification to bring the system in a configuration that satisfies this a priori specification. Given the fact the data set will have several other elements requiring themselves strong changes in the system as well, making them all coexist is much more likely with attractors whose basins

[19]These bounds depend on the problem and reflect what cyclic output is to be considered valid. For instance, when comparing this algorithm with the learning of a data set of cycles of size X with the "out-supervised" learning, then $\texttt{min}_{cs} = \texttt{max}_{cs} = X$. On the other hand, if the only restriction is for the output to be cyclic, then $\texttt{min}_{cs} = 2$, $\texttt{max}_{cs} = K$; where K is taken arbitrarily large. Usually, $\texttt{min}_{cs} = 2$, $\texttt{max}_{cs} = 8$ since it is very rare to naturally find cycles longer than 8 and they are often unstable.

[20]Original means that each pattern composing the limit cycle attractor must be sufficiently different from all other patterns of all $\mathbf{cycle}^{\mu}$. The difference between two patterns is measured as their Hamming distance, usually requiring $d_H > 3$.

are not too wide. This “in-supervised” process requires much smoother changes on the system since the attractors are already there.

The second reason behind this intrinsic robustness to noise comes from the attractors proposed by the system having large and stable basins of attraction. This can be understood by looking at the major difference between the two learning process' uses of the iterative Hebbian learning: while the “out-supervised” algorithm takes a data set and applies the learning process until convergence, the “in-supervised” algorithm only tries to learn the (self-generated) outputs for a small period of time before trying to see how the response of the system has changed. In other words, for an attractor to be present as a response to a stimulus at the end of the learning process, it needs to be strong enough and have a large enough basin of attraction in order to resist those weight modifications during the trial–and–error process of this learning process. Of course this is not a guarantee but neither are the learning noisy versions of the elements in the data set. Those reasons are just indications in favor of the natural robustness of the attractors obtained by this learning algorithm. For this reason, the parameters ln and ln_b have been set to 0 during “in-supervised” learning[22]

The algorithm presented here learns to map external stimuli into the network's cyclic attractors in a very unconstrained way. Provided these attractors are derived from the ones spontaneously proposed by the network, a form of supervision is still needed (how to create an original cycle and how to know if the proposed cycle is original). However, recent neurophysiological observations have shown that in some cases synaptic plasticity is effectively guided by a supervised procedure [Franosch et al., 2005; Gutfreund et al., 2002].

The supervised procedure proposed here to create and to test original cycles has no real biological ground. Nevertheless, whatever supervised procedure is chosen, the part of the study concerned with the capacity of the network and the background chaos remains valid.

5. Conclusion

Observations made by Molter and Bersini [2003a,b] on small recurrent neural networks show that they can be very effective as memory. Experiments over random networks show that, if the mean signal of the system is filtered into a discrete alphabet, for some weight configurations the number of existing unique and robust attractors is huge. More importantly, it clearly appears that there is some correlation between capacity (in term of symbolic robust original attractors) and the spontaneous dynamics of the system. To show that, for each weight configuration, after the capacity has been computed, its global dynamics is quantified as the mean Lyapunov exponent. When this experiment is repeated for a large number of random networks, it appears that, when the system is overall stable, it has a very poor capacity, and a system with large capacity always has a complex background dynamics.

These results were inspiring, and motivated the introduction of a learning algorithm that could store a given data set in a recurrent neural network's dynamics. This thesis first tested approach was the back-propagation through time algorithm, but it performed very poorly for memory-like tasks. This fact has been explained

[22]As shown in the next chapter, this does not reflect badly on the “in-supervised” algorithm, and even more it will be shown that the “in-supervised” learning algorithm seems to provide stronger attractors than the “out-supervised” learning algorithm using learning noise.

by the need of the gradient-based algorithm for a stable network in order to converge and not be constantly perturbed by the bifurcations that can be found in a system with complex dynamics [Bengio et al., 1994; Doya, 1992]. Unfortunately, this also implies very poor capacity.

To work around the limitation of a gradient descent global algorithm, a more biologically plausible approach was proposed. The first solution was the "out-supervised" learning algorithm. This algorithm is an iterative supervised Hebbian learning, and as such works locally without depending on the gradient, which makes it avoid the problem previously mentioned. This algorithm is nothing new but the classical solution has been extended to work with external stimuli.

Finally, a less rigid algorithm was developed, called "in-supervised" learning was proposed. This time, the network learns to map given stimuli to dynamical outputs naturally present in the system. This makes it some half-way solution between the random network and the iterative Hebbian procedure. Even though this algorithm relies on an iterative Hebbian process, it does not need to wait for the Hebbian process to converge, since if the generated output cannot be rapidly learned, the system can propose a new output. The generated output must however respect some constraints. The output must be a periodic cycle with a certain range and it must be a unique sequence for each stimulus. However, it can share some pattern since the external stimuli are different and thus they can provide the necessary information to alleviate the ambiguity.

In conclusion, this chapter introduced several original contributions. First, the study on the BPTT algorithm for memory-like tasks [Molter, Salihoglu, and Bersini, 2004a]. Then, it proposed an improvement over the classical iterative Hebbian associative learning process using an external stimulus layer and using cycles as substrate for information over fixed-point patterns [Molter, Salihoglu, and Bersini, 2007b]. On top of this algorithm, a completely new, less supervised, version was built [Molter, Salihoglu, and Bersini, 2007b].

Algorithm 1 Pseudo code of the "out-supervised" algorithm

```
Require: A data set $\mathcal{G}$ to learn, and a network composed of an associative layer
    and an input layer ($\mathbf{W_R}$ and $\mathbf{W_S}$)
 1. Fill the data set $\mathcal{D}$ with the data to learn ($\mathcal{D}$=$\mathcal{G}$);
 2. A data $\mathcal{D}^\mu$ is chosen randomly from the data set $\mathcal{D}$;

    {learning with no noise}
 3. for idcyc going from 1 to $l_\mu$ do
 4.     The external stimulus is initialized with $\chi^\mu$
        Internal states are initialized with $\varsigma_i^{\mu,\texttt{idcyc}}$;
 5.     One synchronous simulation is performed;
 6.     for all neuron $i$ do
 7.         if $\operatorname{sgn}(x_i) \neq \varsigma_i^{\mu,(\texttt{idcyc}+1) \mod l_\mu}$ then
 8.             all the synaptic connections impinging on this neuron are modified ac-
                cording to the Hebbian learning rule:
(61)            $w_{ij}^R \mapsto w_{ij}^R + \varepsilon_s \varsigma_i^{\mu,(\texttt{idcyc}+1) \mod l_\mu} \varsigma_j^{\mu,\texttt{idcyc}}$
                $w_{is}^S \mapsto w_{is}^S + \varepsilon_b \varsigma_i^{\mu,(\texttt{idcyc}+1) \mod l_\mu} \chi_i^\mu$
 9.             Go back to step 1 and restart with the whole data set.
10.         end if
11.     end for
12. end for
Ensure: Data $\mathcal{D}^\mu$ is stable. {The next steps aim at improving robustness to noise.}

    {learning with noise}
13. The learning noise parameters $ln$ and $ln_b$ are initialized with fixed values.
14. for $n_{test}$ time do
15.     for idcyc going from 1 to $l_\mu$ do
16.         Using $ln_b$, a noisy version $\chi_{ln_b}^\mu$ of $\chi^\mu$ is generated as an external stimulus;
17.         Using $ln$, a noisy version $\varsigma_{ln}^{\mu,\texttt{idcyc}}$ of $\varsigma^{\mu,\texttt{idcyc}}$ is generated as initial internal
            states;
18.         To skip the transient phase, $kl_\mu$ synchronous simulations are performed
            (where $k \in \mathbb{N}^+$ is a parameter of the algorithm);
19.         The internal state is stored: $x_{current}$
            one more simulation is performed: $x_{current} \mapsto x_{next}$.
20.         for all neuron $i$ do
21.             if $\operatorname{sgn}(x_{next,i}) \neq \varsigma_i^{\mu,(\texttt{idcyc}+1) \mod l_\mu}$ then
22.                 all the synaptic connections impinging on this neuron are modified
                    according to the Hebbian learning rule:
(62)                $w_{ij} \mapsto w_{ij} + \varepsilon_s \varsigma_i^{\mu,(\texttt{idcyc}+1) \mod l_\mu} x_{current,j}$
                    $w_{is} \mapsto w_{is} + \varepsilon_b \varsigma_i^{\mu,(\texttt{idcyc}+1) \mod l_\mu} \chi_{ln_b,i}^\mu$
23.                 Go back to step 1 and restart with the whole data set.
24.             end if
25.         end for
26.     end for
27. end for
Ensure: Data $\mathcal{D}^\mu$ is a stable and robust attractor of the network. $\mathcal{D}^\mu$ is removed
    from $\mathcal{D}$ ($\mathcal{D} = \mathcal{D} \backslash \mathcal{D}^\mu$). The whole process is continued from step 2 if $\mathcal{D}$ is not
    empty.
```

Algorithm 2 Pseudo code of the "in-supervised" algorithm

```
1. Simulation of the network
2. for all data $\mathcal{D}^{\mu}_{bm}$ to learn do
3.     the stimulus is initialized with $\chi^{\mu}$;
4.     the states are initialized with $\varsigma^{\mu,1}$ which are obtained from the previous iteration (or randomly the first time);
5.     simulate the network for some steps in order to skip the transient part;
6.     the states $\varsigma^{\mu,1}$ crossed by the network's dynamics are stored in output$^{\mu}$.
7. end for
8. Proposal of an attractor code
9. for all data $\mathcal{D}^{\mu}_{bm}$ to learn do
10.    if output$^{\mu}$ is a cycle of period greater than $\texttt{max}_{cs}$ then
11.        compress the output: output$^{\mu}$ $\mapsto$ cycle$^{\mu}$ (see Algorithm 3)
12.    end if
13.    if output$^{\mu}$ is a cycle of period less than $\texttt{min}_{cs}$ then
14.        expand the output: output$^{\mu}$ $\mapsto$ cycle$^{\mu}$ (see Algorithm 4)
15.    end if
16.    if a pattern contained in cycle$^{\mu}$ is too correlated (the normalized Hamming distance is below 5) with any other pattern then
17.        this pattern is slightly modified to make it "original" (by adding some noise)
18.    end if
19. end for
20. the data set $\mathcal{D}_{am} = (\chi^{\mu}, \textbf{cycle}^{\mu})$ is created.
21. Learning the information
22. using the "out-supervised" algorithm (see Algorithm 1 on the facing page), the data set $\mathcal{D}_{am}$ is tentatively learned during a very limited number of time steps. This step uses the algorithm as described before but does not wait for it to converge.
23. if for any $\mathcal{D}^{\mu}_{bm}$, output$^{\mu}$ is not a valid limit cycle attractor then
24.     Go back to step 2 {An output is valid if it is a cyclic pattern within the required range ($[\texttt{min}_{cs}, \texttt{max}_{cs}]$) }
25. end if
```

Algorithm 3 Pseudo code of the "in-supervised" algorithm: Compression process

```
Require: output$^{\mu}$ = $\{\varsigma^{\mu,1}, \cdots, \varsigma^{\mu,\texttt{max}_{cs}}, \cdots\}$ is a cycle of period greater than $\texttt{max}_{cs}$ (or even non-periodic)
1. cycle$^{\mu}$ is generated by truncating output$^{\mu}$: cycle$^{\mu}$ = $\{\varsigma^{\mu,1}, \cdots, \varsigma^{\mu,p_s}\}$ {$p_s \in [\texttt{min}_{cs}, \texttt{max}_{cs}]$ is a parameter[21] of the algorithm called the "compression period"}
```

Algorithm 4 Pseudo code of the "in-supervised" algorithm: Expansion process

```
Require: output$^{\mu}$ = $\{\varsigma^{\mu,1}, \cdots, \varsigma^{\mu,q}\}$ with $q < \texttt{min}_{cs}$
1. cycle$^{\mu}$ is generated by duplicating output$^{\mu}$ until $|\textbf{cycle}^{\mu}| = p_e$: cycle$^{\mu}$ = $\{\varsigma^{\mu,1}, \cdots, \varsigma^{\mu,q}, \varsigma^{\mu,q+1}, \cdots \varsigma^{\mu,p_e}\}$, where $\varsigma^{\mu,q+i} = \varsigma^{\mu,(i-1 \mod q)+1} + \kappa_i$ {$p_e \in [\texttt{min}_{cs}, \texttt{max}_{cs}]$ is a parameter of the algorithm called the "expansion period", and $\kappa_i$ is some randomization applied to the patter $q+i$}
2. In order to guarantee the originality of the newly generated cycle, each duplicated pattern is slightly modified (with a random noise).
```

CHAPTER 5

Capacitive and dynamical results

This chapter shows some experimental results obtained with the two iterative Hebbian learning algorithms presented in the previous chapter. Two main aspects of these procedures are explored: their capacity, the dynamics the system produced after such a learning, and the benefits one can make out of it. The first section takes a closer look at the encoding capacity obtained with such learning procedures, the second section covers dynamical properties of the system and finally the last section explores added features of chaotic dynamics in a memory-like system. None of the experiments aims to be biologically accurate but they all are designed to highlight some important features of the model and the benefits of such complex dynamics. This chapter works with very simple instances of the model defined in Chapter 3 on page 73: all results in this section were obtained without context layer and global inhibitor ($\mathbf{W_C} = \mathbf{W_V} = \mathbf{W_A} = \mathbf{W_I} = 0$). Hence, the matrix weight $\mathbf{W}$ becomes :

$$\mathbf{W} = \begin{pmatrix} \mathbf{W_R} & 0 & \mathbf{W_S} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{69}$$

The model is left with only the two main components: the associative layer and the input layer that serves to model the incoming stimuli, as shown in the Figure 5.1 on the next page.

1. Encoding Capacity

1.1. "Out-supervised" learning. The goal here is to establish what kind of capacitive performance the "out-supervised" algorithm can achieve. But capacity without robustness is meaningless; so in order to accurately measure this capacity two noise parameters are introduced: nu, which is the noise injected[1] in one of the learned cycle's patterns ($\varsigma_{nu}^{\mu,\mathtt{idcyc}}$), and nu_b, which is the noise injected in the external stimuli ($\chi^{\mu}_{nu_b}$).

After injecting noise to the system, specified by the couple (nu, nu_b), robustness is measured by evaluating how well the dynamics is able to retrieve the original stored pattern (in case of a cycle, all the steps are evaluated). This measure can be done using the Hamming distance between two patterns (see Equation 58 on page 92) or using the overlap m^{μ} between two patterns, which is given by:

$$m^{\mu} = \frac{1}{N}\sum_{i=1}^{N} \varsigma_i^{\mu,\mathtt{idcyc}}\ \varsigma_{i,\mathtt{noisy}}^{\mu,\mathtt{idcyc}} \tag{70}$$

[1]Noise injection is done as described in the previous Chapter.

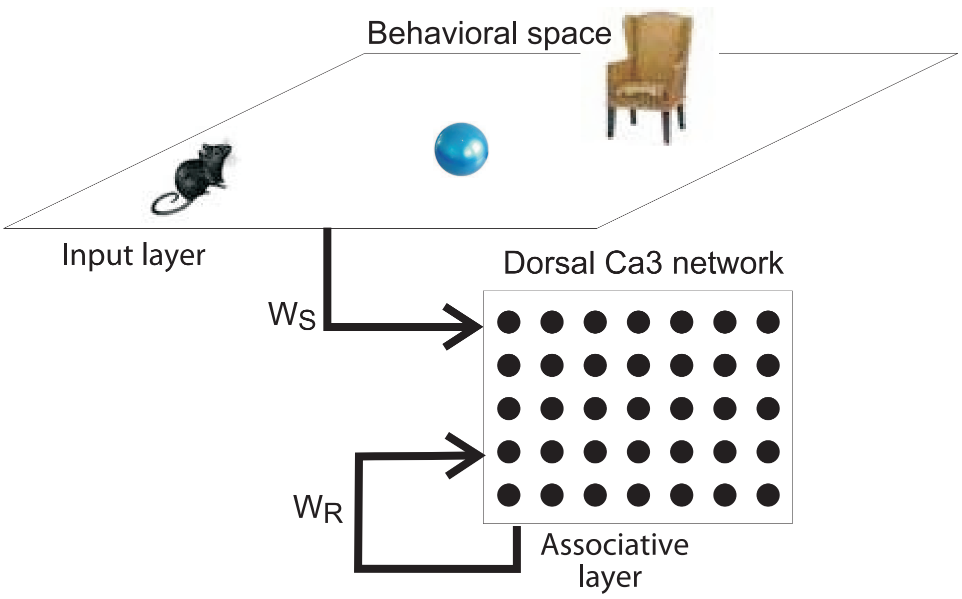


FIGURE 5.1. Limited architecture of the model, as it is used in this chapter, with only the active components.

Two patterns which match perfectly have an overlap equal to 1, while it is −1 in the opposite case. The overlap m and the Hamming distance d_H are related by:

$$d_H = N\,\frac{m+1}{2}, \qquad \text{where } N \text{ is the number of neurons} \tag{71}$$

The use of the overlaps allows the direct use of results from [Forrest and Wallace, 1995], where the fraction of properly recalled nominal states is plotted as a function of increasing initial[2] overlaps m_0^μ.

1.1.1. *Auto-associative memory.* The first tests use the "out-supervised" algorithm to learn patterns in an auto-associative way: results are shown in Figure 5.2 on the next page. Two networks are tested here. First the classical (Hopfield like) network with no external stimulus (dotted curves in the figure). The data set for this case is described in equation 51 on page 89, data are generated randomly, with an uniform distribution, in $\{-1, 1\}$. The second network has an external stimulus (plain curve on the figure) and a data set characterized by the equation 52 on page 89 also generated randomly, with an uniform distribution, in $\{-1, 1\}$.

The comparison is done for content-addressability. For this, after learning, the network is fed with patterns to recall (with increasing noise) and the result is the percentage of successful recalls. When testing with external stimulus, the pattern also continuously feeds the external stimulus.

Two results can be highlighted. First, as shown by Forrest and Wallace [1995], learning noise clearly increases content-addressability. Secondly, the external stimuli clearly increase the content-addressability of the system. Since the same noisy pattern feeds the external stimulus and initializes the internal state[3], this curve can be straightforwardly compared to the other curves obtained without external stimuli, which for example indicates close to 20% more correct recalls for $m_0 = 0.4$, when using the external stimulus.

[2]The initial overlap is the one that can be measured before any simulation.

[3]This can be formalized as $\xi^\mu_{nu_b} = \xi^\mu_{nu}$ or as $nu_b = nu$.

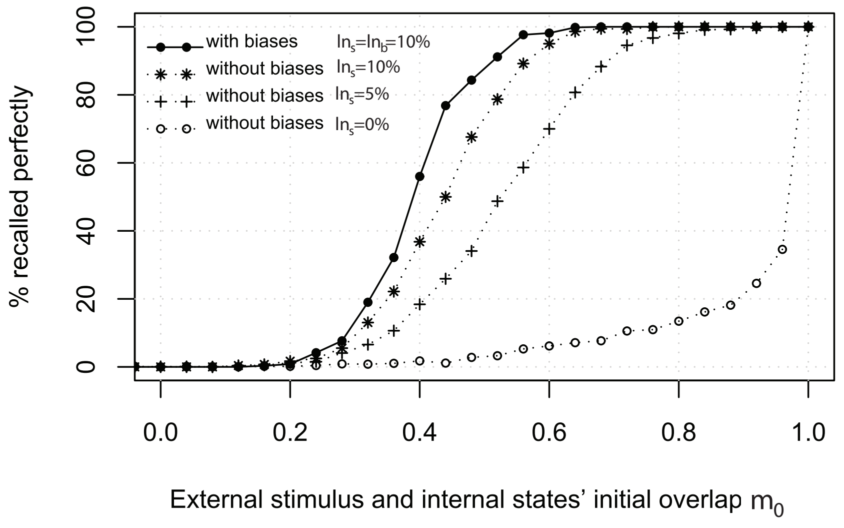


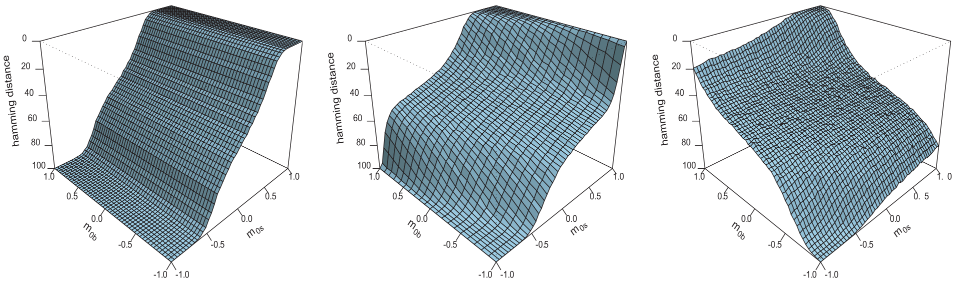
FIGURE 5.2. Percentage of correct recalls with the "out-supervised" algorithm. The recall success rate is ploted in function of the noise injected on the initial pattern. The dotted curves show results for different learning noises with no external stimulus. The plain curve shows the same data set learned using external stimulus, which seems to clearly increase content-addressability. Here the network has 100 neurons and has learned 25 patterns ($\alpha = 0.25$).

One drawback of the external stimulus is the increase in the number of parameters of the system. Learning noise (ln) and learning rate (ε) have been duplicated with their counterpart for the stimulus (ln_b and ε_b). The values those parameters should have is not straightforwardly defined and need to be adjusted depending on the problem (in other words depending on the expected noise after learning, nu and nu_b).

ε_s	0.05	It is chosen quite small in order to avoid brutal changes. From experimentation, a smaller value slows the learning process too much, and a larger value induces too strong changes.
ε_b	0.05	It is kept similar to the inner state's learning rate.
ln_s	$*$	The learning noise is shown on the plots.
ln_b	$*$	The learning noise is shown on the plots.
l_μ	1	These experiments deal with fixed-point attractors.
k	10	Used to compute the size of the transient phase to skip.
$\|A\|$	100	Size of the associative layer.
$\|S\|$	100	Size of the input layer.

TABLE 1. This table recaps all the parameters used for the experiments in Section 1.1.1 on the facing page with their value. A value shown as $*$ indicates that this variable takes different values during the experiment, and those values are reported on the corresponding figure.

1.1.2. *Hetero-associative memory.* Once the system has external stimuli, it also has the possibility to work as a hetero-associative memory. This means that, this time, the information on the internal states of the network and the information on the external stimuli will be different. The data set for the learning procedure is formalized in equation 53 on page 89.



(a) without stimuli: ln_s=12 (b) with stimuli: ln_s=14,ln_b=6 (c) with stimuli: ln=15,ln_b=0

FIGURE 5.3. Content-addressability of the hetero-associative system (measured on the normalized Hamming distance) with varying noise on the external stimulus (m_{0b}) and the internal states (m_{0s}). The difference between the 3 plots arises from the learning noise parameters. The size of the layers is set to $|A| = |S| = 100$, and the load is set to $\alpha = 0.25$.

Figure 5.3 shows the noise robustness, as a normalized Hamming distance between stored and recalled patterns. The recall quality has been tested for different noise parameters on the stimulus and the internal states. On the plots, the overlap between the given pattern and the correct one indicates the noise. And it ranges from perfect pattern ($m_{0s} = 1$) to the opposite pattern ($m_{0s} = -1$). Each plot shows the result for different learning noise parameters on a 100 neurons network ($|A| = |S| = 100$) which has to learn 25 patterns ($\alpha = 0.25$).

The first plot (Fig. 5.3(a)) is used as a reference case and is learned without external stimuli (all weight are set to zero: $\forall i, s \colon w_{is} = 0$). This first test is nothing more than the auto-associative example with no input, described above; and the invariant response to noise on the stimulus ($m_{0b} < 1$) reflects that. The other two plots show the parameter tuning's importance. If during learning no noise is used on the external stimuli ($ln_b = 0$, Fig. 5.3(c)), while noise is injected to the internal states ($ln = 15$), the external stimuli are treated with great confidence by the system. As a result, if the external stimulus is perfectly given ($m_{0b} = 1$), the stored pattern is almost always recovered: in the worst case (when initial states are randomly chosen, i.e. $m_{0s} = 0$), in average only 15% of error is obtained in the recovered attractor. However, if external stimuli are noisy the performance decreases fast.

On the other hand, if noise is applied to the external stimuli as well during learning (Fig. 5.3(b)) then the network is more tolerant to noise on the stimulus during test but is also much more cautious with the information present on the external stimulus when compared to the case with no noise on the external stimulus (Fig. 5.3(c)). This leads to a less successful recall quality than when there was no noise on the input.

The first two plots (Fig. 5.3(a),(b)) look more related, but a closer look reveals that, naturally, the first plot is impervious to noise on the external stimulus (since it is disabled) but it also shows that the presence of external stimuli helps the content-addressability unless the stimulus suffers heavy noise ($m_{0b} < 0.0$).

The hetero-associative task may seem dull and better achieved with a feed forward network, since doing such hetero-associative tasks is the basic functionality of these networks and they were built for it. However, a major difference justifies the use of recurrent neural networks. In feed forward networks, one given input is associated with a specific output; here the output is the result of the input but also of the internal dynamics of the system. This allows the system to learn hetero-associative data sets where two identical external stimuli lead to different outputs because of their initial internal states' values (see Figure 5.4). This can be seen as a contextual hetero-associative process.

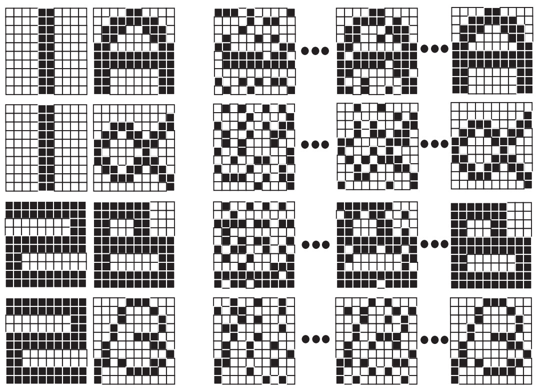


FIGURE 5.4. Context dependent data set. Here, the network has learned a data set that requires contextual information in order to be processed and that cannot be done with a simple feed forward association. The data set (shown on the left side) is composed of four elements: $\mathcal{D} = \{(1, A), (1, \alpha), (2, B), (2, \beta)\}$. The right side shows the evolution of the internal states after having initialized the external stimulus with the corresponding input and the internal state with a noisy version of the expected output. Thanks to the internal state of the system these networks can describe a multi-valued hetero-associative functions, while perceptrons are limited to single-valued functions (where each stimulus gives a different output).

1.1.3. *Learning Sequences of patterns.* The next logical step is to test this algorithm for learning sequences of patterns (each one indexed by an external stimulus). The network's capacity for learning sequences of various lengths in absence of noise can be seen in Figure 5.5 on the next page. The load parameter fails short when dealing with a sequence of inputs since it clearly requires more storage capacity to store a sequence of 10 elements than two fixed points. To take this into account, a generalized load parameter α_r is defined. If q_r is the total number of patterns in all the sequences, then the generalized load parameter can be defined as: $\alpha_r = q_r/N$. From the results in Figure 5.5 on the following page, it appears that this load is constant ($\alpha \sim 1.5$), showing that the maximum capacity is independent of the size of the sequence to learn. The addition of an external stimulus pushes the maximum load to $\alpha \sim 1.6$.

The external stimuli are not just there because they increase the capacity of the system; they also greatly help for the quality of the recalls. Figure 5.6 on page 107 shows the same content-addressability tests that were performed with

ε_s	0.05	It is chosen quite small in order to avoid brutal changes. From experimentation, a smaller value slows the learning process too much, and a larger value induces too strong changes.
ε_b	0.05	It is kept similar to the inner state's learning rate.
ln_s	$*$	The learning noise is shown on the plots.
ln_b	$*$	The learning noise is shown on the plots.
l_μ	1	These experiments deal with fixed-point attractors.
k	10	Used to compute the size of the transient phase to skip.
$\|A\|$	100	Size of the associative layer.
$\|S\|$	100	Size of the input layer.

TABLE 2. This table recaps all the parameters used for the experiments in Section 1.1.2 on page 103 with their value. A value shown as $*$ indicates that this variable takes different values during the experiment, and those values are reported on the corresponding figure.

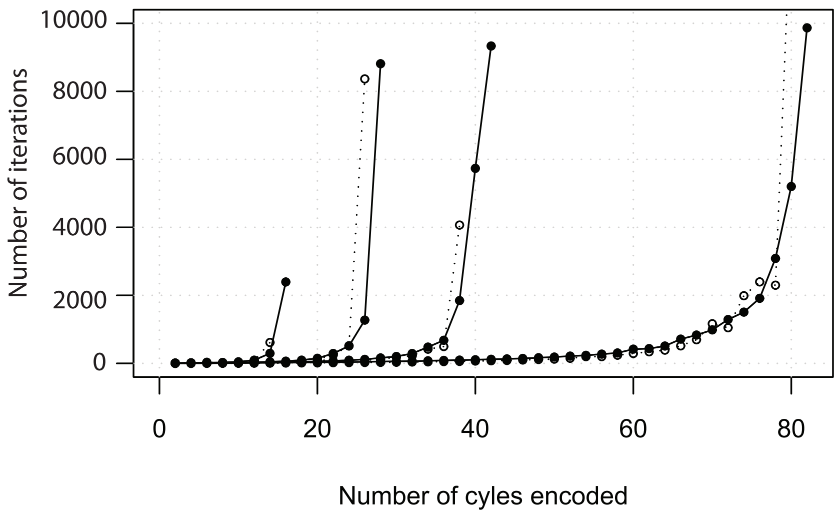


FIGURE 5.5. Capacity of the "out-supervised" algorithm. Measured for learning sequences of various size. The numbers of iterations needed to store an increasing number of sequences have been plotted. The plot shows from left to right the capacity for sequences of length 10, 6, 4 and 2. The dotted line shows the same result without external stimulus, and again the stimulus seems to help. The size of the layers is set to $|A| = |S| = 100$.

hetero-associative learning (Figure 5.3 on page 104) and the conclusion is pretty much the same. When learning without noise on the external stimuli (sub figure (c)), the content-addressability is very good as long as the trusted external stimulus is noise free. The sub figures (a) and (b) respectively show the results without and with external stimuli and the addition clearly helps the recall quality.

As with the hetero-associative systems, the external stimuli's enhancement goes beyond numerical results and adds new possibilities to the memory. It is possible, due to the disambiguation they provide, to learn different sequences that contain the same pattern. Since the system is deterministic, this would be extremely hard without external stimulus. Without external stimulus the system needs to find two

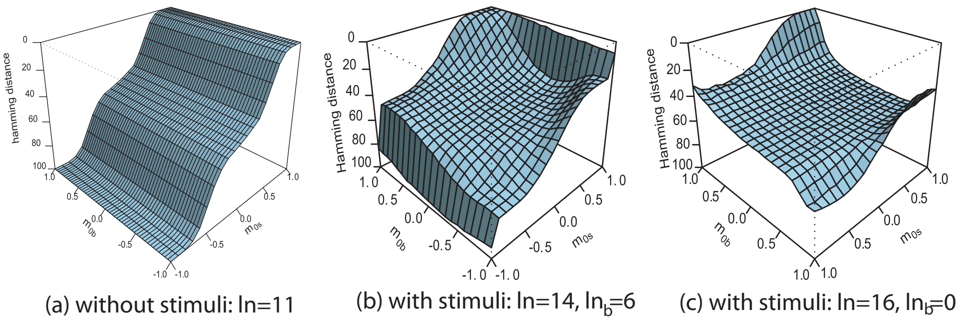


FIGURE 5.6. Noise tolerance of the "out-supervised" learning. Hamming distance between the expected sequences of patterns and the ones obtained is plotted in function of the noise injected both in the initial states (m_{0s}) and in the external stimulus (m_{0b}). Each plot has been obtained for different values of the noise parameters during learning and the left plot gives the results without external stimulus for comparison. The size of the layers is set to $|A| = |S| = 100$, and the load parameters are set to $\alpha_r = 0.25$ and $\alpha = 0.05$ (for these tests, the data set was composed of 5 cycles of size 5).

stable attractors which both at some point in their trajectory are very close to each other[4]. Intuitively, it is easy to see the problem here: two very close points in space must take very different trajectories; unfortunately the easiest way to do so is to be in a chaotic region of the system, which in turn does not satisfy the criteria of being a stable cyclic trajectory. Figure 5.7 on the next page shows an example of such a data set.

ε_s	0.05	It is chosen quite small in order to avoid brutal changes. From experimentation, a smaller value slows the learning process too much, and a larger value induces too strong changes.
ε_b	0.05	It is kept similar to the inner state's learning rate.
ln_s	$*$	The learning noise is shown on the plots.
ln_b	$*$	The learning noise is shown on the plots.
l_μ	$*$	These experiments deal with various size cyclic attractors.
k	10	Used to compute the size of the transient phase to skip.
$\lvert A\rvert$	100	Size of the associative layer.
$\lvert S\rvert$	100	Size of the input layer.

TABLE 3. This table recaps all the parameters used for the experiments in Section 1.1.3 on page 105 with their value. A value shown as $*$ indicates that this variable takes different values during the experiment, and those values are reported on the corresponding figure.

1.2. "In-supervised" learning. The goal of this part is to perform the same kinds of tests with the "in-supervised" learning. Again, the main interests are the capacity and the content-addressability of the data the networks learn. The results are compared with the ones obtained for "out-supervised" learning.

[4]Since the output of the system is filtered, they can be very close (yet different) but still have the same filtered output.

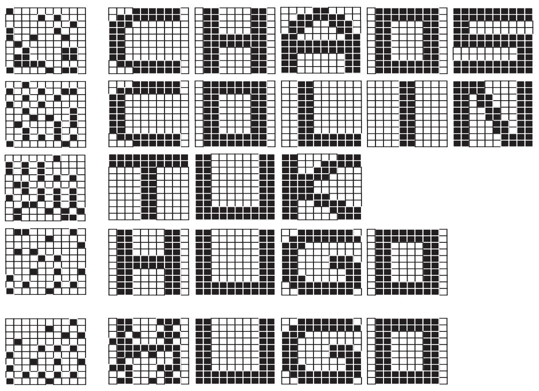


FIGURE 5.7. A data set with (cyclic) sequence sharing patterns. The data set maps a randomly generated external stimulus to $\{chaos, colin, tuk, hugo\}$. In this particular data set the pattern "C" is expected to go to "H" or "O" depending on the external stimulus, "O" is expected to go to "S","L" or "H","U" is expected to go to "K" or "G", and "H" is expected to go to "A" or "U". The four first rows correspond to the data to learn: the first image is the external stimulus while the following ones compose the sequence. The last row shows the network in action: noise is both added on the external stimulus and the states of the last data. Nevertheless, the sequence is perfectly recovered.

1.2.1. *Performance of the algorithm.* The first test (see Figure 5.8 on the next page) to put the two algorithms in perspective is their efficiency. At the end of the day, when the learning is said to have failed to learn a data set, it is often because even if it was possible for the system to converge at some point it would have taken too much time to wait for it. In Figure 5.5 on page 106, this phenomenon is clearly observable. Here, both algorithms are asked to learn a data set of increasing size. The performance is measured in term of the number of iterations the algorithm needs in order to learn the given data set, where an iteration is defined as one hundred weight modifications defined in Equation 60 on page 92. The data set of the "out-supervised" algorithm is composed of stimuli associated with period-2 cycles, and to be fair the "in-supervised" algorithm is limited to work with only period-2 cycles as well ($\mathtt{min}_{cs} = \mathtt{max}_{cs} = p_e = p_s = 2$).

This plot also shows the capacitive limits of both algorithms; in order to show the maximum encoding capacities, two parameters enforcing the content-addressability (ln_s and ln_b) have been set to 0. As expected, the "in-supervised" version, by letting the network to decide how to map the stimuli, outperforms by far its "out-supervised" counterpart, where all the mappings are specified before the learning. This performance could be even better if the limitation on the cyclic output of the "in-supervised" algorithm were to be removed. It is also important to note that the parameters ln_s and ln_b play a big role in the content-addressability for the "out-supervised" algorithm, but the "in-supervised" algorithm produces very good results even without having to explicitly guarantee noise resistance (see section 4.2 on page 95). In fact all the results presented below are done without specific noise during learning for the "in-supervised" algorithm.

1.2.2. *Content-Addressability.* The series of plots in Figure 5.9 on page 111 compare the robustness to noise of networks after learning different data sets with

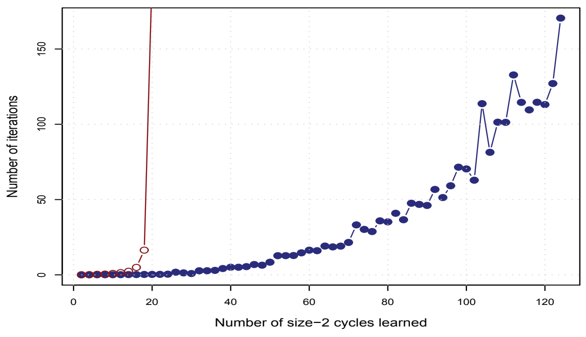


FIGURE 5.8. Performance of the two learning algorithms. The "in-supervised" (blue curve) and "out-supervised" (red curve) algorithms are compared. The x-axis shows the size of the data set composed of period-2 cycles. The y-axis shows the number of iterations (=100 weight modifications) needed to learn the given data set. Incidentally, this plot also shows their respective capacitive limit. The size of the layers is set to $|A| = |S| = 25$, and no learning noise ($ln_s = ln_b = 0$). The oscillation observed is statistical and can be reduced by averaging over multiple runs.

ε_s	0.05	It is chosen quite small in order to avoid brutal changes. From experimentation, a smaller value slows the learning process too much, and a larger value induces too strong changes.
ε_b	0.05	It is kept similar to the inner state's learning rate.
ln_s	0	The learning noise is null in order to keep the test fair for the "out-supervised" learning algorithm
ln_b	0	The learning noise is null in order to keep the test fair for the "out-supervised" learning algorithm
l_μ	2	These experiments deal with size-2 cyclic attractors.
$\mathtt{min}_{cs}$	2	Minimum and maximum cycle size are set equals to be fair to the "out-supervised" learning in the comparison.
$\mathtt{max}_{cs}$	2	Minimum and maximum cycle size are set equals to be fair to the "out-supervised" learning in the comparison.
p_s	2	Preferred compression period must be in $[\mathtt{min}_{cs}, \mathtt{max}_{cs}]$.
p_e	2	Preferred expansion period must be in $[\mathtt{min}_{cs}, \mathtt{max}_{cs}]$.
κ_i	3	Size of the input layer.
k	10	Used to compute the size of the transient phase to skip.
$\|A\|$	25	Size of the associative layer.
$\|S\|$	25	Size of the input layer.

TABLE 4. This table recaps all the parameters used for the experiments in Section 1.2.1 on the preceding page with their value.

both algorithms. The quality of the output is measured as always with the normalized Hamming distance between the stored sequence and the recovered sequence in function of the noise injected in the external stimulus (m_{0b}) and in the initial states (m_{0s}). The "in-supervised" algorithm is again a considerable improvement over its counterpart. The noise is represented as a Hamming distance. In order to generate a noisy pattern with an Hamming distance d from the original one a uniform random sampling without re-pick is performed and d units are selected and their value is inversed (multiplied by -1).

Comparing plots (c) and (d) in Figure 5.9 on the next page clearly shows the difference between the two algorithms after learning 25 period-4 cycles. While the sequences stored with "in-supervised" are very robust to noise, they no longer have any content-addressability after learning with the "out-supervised" algorithm: any amount of noise corrupts the output beyond repair (there is no correlation whatsoever between the output and the expected output). This figure also shows that, in the "in-supervised" case, the external stimuli play a stronger role in indexing the stored data.

ε_s	0.05	It is chosen quite small in order to avoid brutal changes. From experimentation, a smaller value slows the learning process too much, and a larger value induces too strong changes.
ε_b	0.05	It is kept similar to the inner state's learning rate.
ln_s	0	The learning noise is null in order to highlight the base noise tolerance of the algorithms
ln_b	0	The learning noise is null in order to highlight the base noise tolerance of the algorithms
l_μ	4	These experiments deal with size-4 cyclic attractors.
$\texttt{min}_{cs}$	4	Minimum and maximum cycle sizes are set equals to be fair to the "out-supervised" learning in the comparison.
$\texttt{max}_{cs}$	4	Minimum and maximum cycle sizes are set equals to be fair to the "out-supervised" learning in the comparison.
p_s	4	Preferred compression period must be in $[\texttt{min}_{cs}, \texttt{max}_{cs}]$.
p_e	4	Preferred expansion period must be in $[\texttt{min}_{cs}, \texttt{max}_{cs}]$.
κ_i	3	Size of the input layer.
k	10	Used to compute the size of the transient phase to skip.
$\|A\|$	100	Size of the associative layer.
$\|S\|$	100	Size of the input layer.

TABLE 5. This table recaps all the parameters used for the experiments in Section 1.2.2 on page 108 with their value.

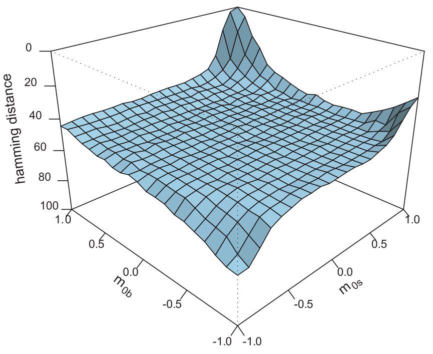
2. Complex dynamics in Hebbian Memories

In the last section, numerical results have been computed assessing both the encoding capacities and the tolerance to noise of these algorithms. Here, the goal is to study the underlying dynamics of a network after learning took place with one of the Hebbian algorithms proposed in Chapter 4 on page 81.

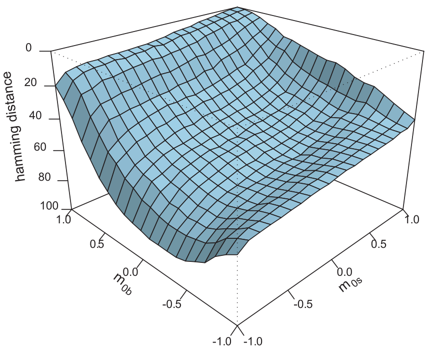
The learning task directly affects the dynamical response one can expect from the network. While learning static patterns stabilizes the network, the coding of information in robust cyclic attractors increases chaos in the network as a background regime, erratically itinerating among brief appearances of these attractors.

Qualitative analyzes are also performed in order to identify the nature of the obtained chaotic dynamics. This is done with classical tools such as the return maps, power spectra and Lyapunov spectra.

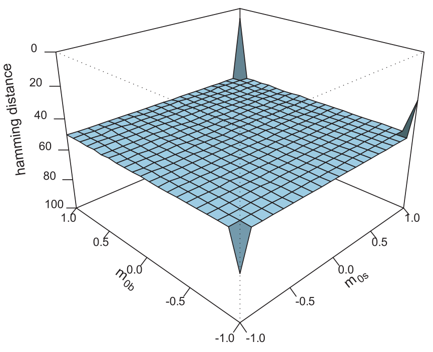
2.1. "Out-supervised" learning. Quantitative analyses of the dynamics encountered in learned networks have been performed using two kinds of measures:



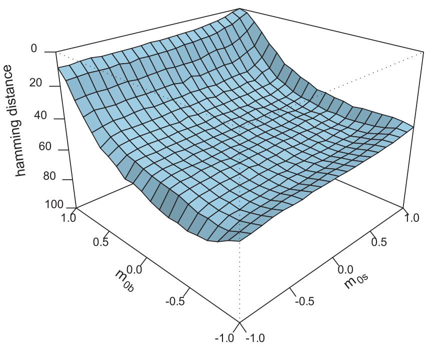
(a) “out-supervised : 12 period-4 cycles



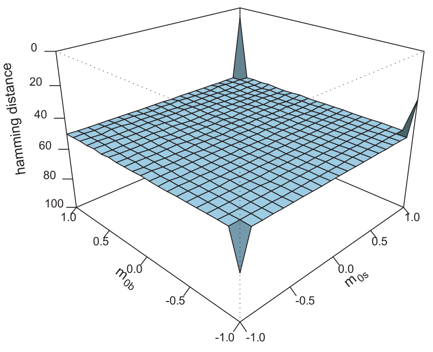
(b) “in-supervised : 12 period-4 cycles



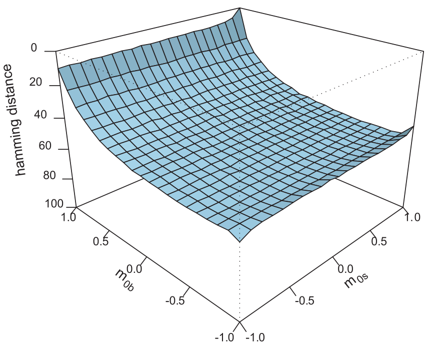
(c) “out-supervised : 25 period-4 cycles



(d) “in-supervised : 25 period-4 cycles



(e) “out-supervised : 37 period-4 cycles



(f) “in-supervised : 37 period-4 cycles

FIGURE 5.9. Noise tolerance of both algorithms. The content-addressability of the network is checked after learning different numbers of period-4 cycles. The plots on the left show the results obtained with the “out-supervised” algorithm, and the ones on the right shows the result obtained with the “in-supervised” algorithm. As for the previous plots, the content-addressability is measured as the normalized Hamming distance between the system's output and the expected output in function of both the initial overlap of the initial states pattern (m_{0s}) and of the external stimulus (m_{0b}). The size of the layers is set to $|A| = |S| = 100$.

the mean Lyapunov exponent[5] and the probability to have chaotic dynamics. Both results are based on statistics over large numbers of learned networks (Here, 1000 networks were tested). For each network, dynamics (obtained by randomly varying the external stimuli and the initial states) have been tested (1000 different configurations). These tests aimed to analyze the so-called background (or spontaneous) dynamics of the network, which can be observed by simulating the system with external stimuli and initial states different from the learned ones. For each simulation the Lyapunov exponent is computed, and dynamics is said to be chaotic if the Lyapunov exponent is greater than a given value slightly above zero, allowing to distinguish chaotic dynamics from quasi-periodic regimes[6].

In order to validate these tests, they are also done over random networks (also called "surrogate networks"). These extra tests highlight the difference between the features specific to the learning processes and those that are present in any network with similar configurations. These random networks have some constraint in order for them to be as close as possible to the kind of network obtained after learning. To this end their weights distribution has to respect the same mean (μ) and same standard deviation (σ):

$$\begin{array}{lll} \mu(w_{ij}^L) = \mu(w_{ij}^S) & \mu(w_{bi}^L) = \mu(w_{bi}^S) & \mu(w_{ii}^L) = \mu(w_{ii}^S) \\ \sigma(w_{ij}^L) = \sigma(w_{ij}^S) & \sigma(w_{bi}^L) = \sigma(w_{bi}^S) & \sigma(w_{ii}^L) = \sigma(w_{ii}^S) \end{array} \tag{72}$$

Thanks to these surrogate networks, it is possible to study the impact of the weights distribution and identify what is the contribution of the learning algorithm and what is naturally present with certain weights configurations. The normal distribution was used for these random networks since it is the one that fits the best the natural weight distribution of the network after learning [Molter, 2005].

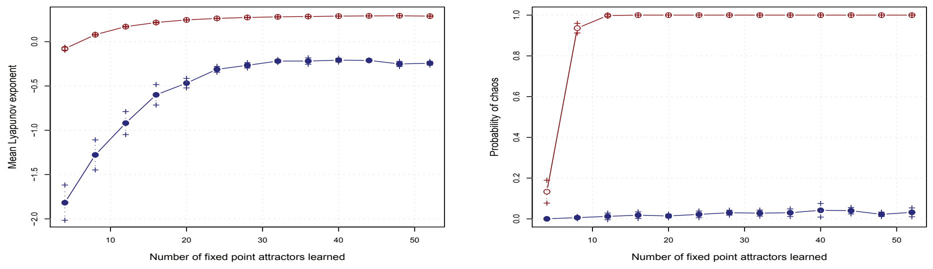
2.1.1. *Spontaneous regime after fixed-point learning.* The next experiments analyze the dynamics of the system after a fixed point attractor data set is learned (see Equation 53 on page 89). When dealing with fixed points, adding noise during learning is mandatory, otherwise the best solution for the system is to converge to the identity weight matrix that can recall all the patterns but with no content-addressability.

Figure 5.10 on the next page shows the mean Lyapunov exponents and probabilities to have chaos for networks after learning. From those results, the following observations can be made:

- The mean Lyapunov exponent for the network after learning is always negative. Even though the mean Lyapunov increases with the size of the data set, it also clearly seems to have an upper limit, which is still negative.
- Chaotic dynamics are very rare and the learning process does not help in any way.
- Surrogate networks show very different results, the probability to find chaos is near 1 and the mean Lyapunov exponent is positive. Those results indicate that complex dynamics are present and cover most of the state space. This indicates that the learning task is responsible for this stabilization of the network.

[5]The computation of the first Lyapunov exponent is done empirically by computing the evolution of a tiny perturbation (re-normalized at each time step) performed on an attractor state [see Albers et al., 1998; Wolf et al., 1984, for more details].

[6]In practice: the dynamics is considered as chaotic if the Lyapunov exponent is greater than 0.01, to avoid confussion with quasi-periodic orbits whose Lyapunov is near zero.



(a) Mean Lyapunov exponent (b) Probability to have chaos

FIGURE 5.10. Dynamics of the network after an "out-supervised" learning. The mean Lyapunov exponent and the probability of chaos are computed for networks after "out-supervised" learning has trained them to learn fixed point attractors. This experiment is repeated over 1000 networks and the results are plotted as the mean and standard deviation (in blue). The results are compared with surrogate networks with the same weight distribution (in red). Hebbian learning of fixed point clearly stabilizes the network. The size of the layers is set to $|A| = |S| = 100$, and the learning noises are set to $ln_s = ln_b = 6\%$.

The "out-supervised" learning algorithm, when working with static patterns, is likely to keep all connections approximately symmetric, so preserving the stability of the learned networks, which is no longer the case for random networks.

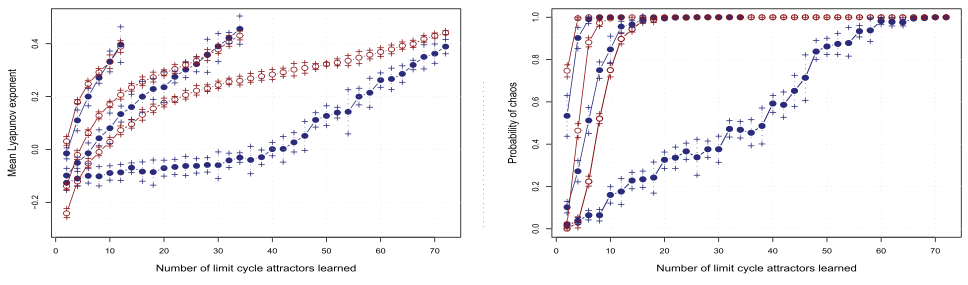
ε_s	0.05	It is chosen quite small in order to avoid brutal changes. From experimentation, a smaller value slows the learning process too much, and a larger value induces too strong changes.
ε_b	0.05	It is kept similar to the inner state's learning rate.
ln_s	0.06	An average learning noise is used
ln_b	0.06	An average learning noise is used
l_μ	1	These experiments deal with fixed-point attractors.
k	10	Used to compute the size of the transient phase to skip.
$\|A\|$	100	Size of the associative layer.
$\|S\|$	100	Size of the input layer.

TABLE 6. This table recaps all the parameters used for the experiments in Section 2.1.1 on the facing page with their value.

2.1.2. *Spontaneous regime after sequence learning.* The second experiment does the same analysis but this time after the system has learned sequences, which leads to diametrically opposed results. Figure 5.11 on the next page shows this analysis done with a increasing number of cycles for a given size, and Figure 5.12 on the following page shows the results for one cycle of increasing size.

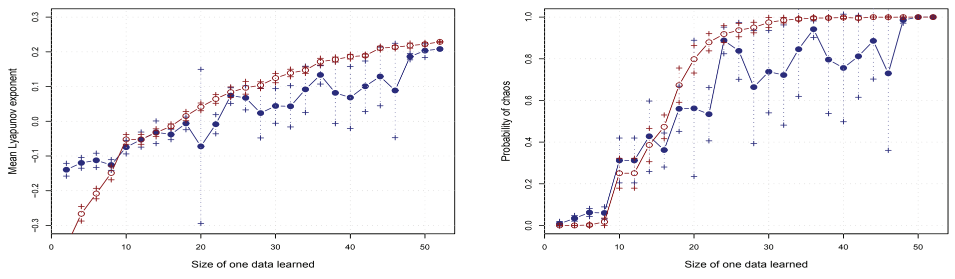
Both experiments here lead to the same observation:

- When the load of the system increases ($\alpha_r \nearrow$), the spontaneous dynamics of learned networks become more and more chaotic, and chaos prevails as the natural spontaneous regime for the network.
- The obtained dynamics are more structured than surrogate networks since the network has at least stored the data set. This is detailed in the next section with the analysis of the type of chaos observed on these networks.



(a) Mean Lyapunov exponent (b) Probability to have chaos

FIGURE 5.11. Dynamical properties after "out-supervised" learning of multiple cycles. Shown as the mean and standard deviation of the dynamical properties of 1000 networks after "out-supervised" learning (in blue). The data sets from left to right encode cycles of period 2, 4 and 10. The surrogate networks are in red. Chaos arises with the increase in the size of the data set that is learned. The size of the layers is set to $|A| = |S| = 100$, and there is no learning noise ($ln_s = ln_b = 0$).



(a) Mean Lyapunov exponent (b) Probability to have chaos

FIGURE 5.12. Dynamical properties after "out-supervised" learning of one big cycle. Shown as in the mean and standard deviation of the dynamical properties of 1000 networks after "out-supervised" learning (in blue). Here the data set contains only one cycle and complexity is brought by increasing its size. The same type of results can be observed. The size of the layers are set to $|A| = |S| = 100$, and there is no learning noise ($ln_s = ln_b = 0$).

- At first the surrogate network's dynamics seems to be very different from the one obtained with learning but, after the load has increased enough, this difference fades away.
- The standard deviation on learned data set is larger than for the surrogate networks. This is understandable, since the learning policy does not try to have a complex dynamics but just learns the data set and it is possible that this system converges to a very stable network. If looked closely, the curve seems to follow a line and each time one sampling is below the expected average on this virtual line the standard deviation becomes large. This is most likely due to some of the learnings to fail and become very stable, thus perturbing the plot. The surrogate networks reflect the general dynamics available under certain weight configurations and thus is less likely to find one of those drastically different networks.

The dynamics observed in the system are very strong chaos (high mean Lyapunov exponents) and seem to cover the whole state space (probability of chaos ~ 1). This kind of dynamics is ergodic[7] [Fusi, 2002]. This looks very similar to the "blackout catastrophe" observed in fixed point attractors networks [Amit, 1989]. Here this blackout arises with a progressive increase of the chaotic dynamics.

When learning cycles, the network is prevented from stabilizing in fixed-point attractors. The more cycles to learn, the more the network is externally constrained and the more the regime turns out to be spontaneously chaotic. This learning process can be seen as a road to chaos using Hebbian learning (see Figure 5.13 on the following page). Here, the graduation is not done on a control parameter (as with the classical roads to chaos) but by an external mechanism simultaneously modifying a set of parameters to fulfill an encoding task. This is why, in Figure 5.13 on the next page, the x-axis shows the learning step of the algorithm responsible for this parameter modification.

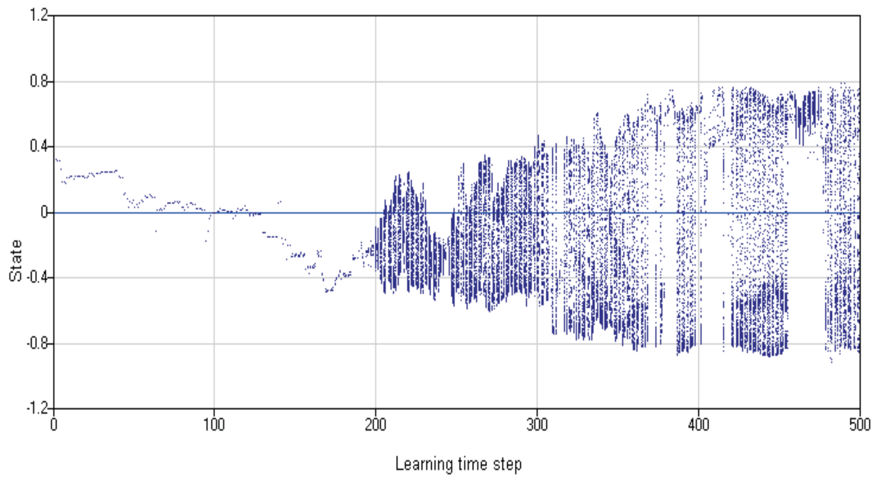
ε_s	0.05	It is chosen quite small in order to avoid brutal changes. From experimentation, a smaller value slows the learning process too much, and a larger value induces too strong changes.
ε_b	0.05	It is kept similar to the inner state's learning rate.
ln_s	0	The learning noise is not needed for these experiments.
ln_b	0	The learning noise is not needed for these experiments.
l_μ	$*$	These experiments deal with various size cyclic attractors.
k	10	Used to compute the size of the transient phase to skip.
$\|A\|$	$*$	Size of the associative layer.
$\|S\|$	$*$	Size of the input layer.

TABLE 7. This table recaps all the parameters used for the experiments in Section 2.1.2 on page 113 with their value. A value shown as $*$ indicates that this variable takes different values during the experiment, and those values are reported on the corresponding figure.

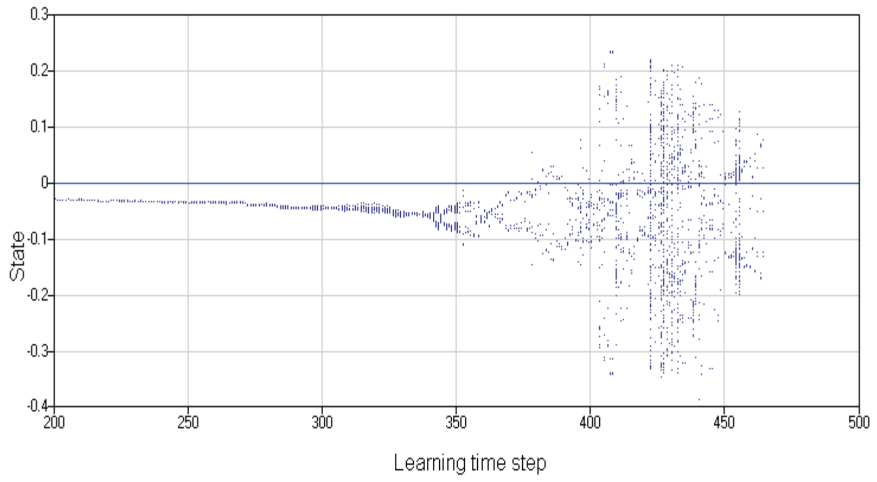
2.1.3. *Nature of chaos.* The final step before proceeding with results obtained with the "in-supervised" algorithm concerns the characterization of the chaotic dynamics. To this end, a qualitative approach is taken, with the use of tools such as the return maps and power spectrum analyses. Three types of chaotic regimes have been identified in these networks. Figure 5.14 on page 118 shows them with: typical return maps of the system, return map of a neuron, and power spectrum. These three types of chaos are identified as:

- **white noise**: Shown in Figure 5.14(a), this dynamics has a characteristic power spectrum with no structure (very similar to white noise). The return map is completely filled, with a bigger density for points at the edges, indicating the presence of saturation. No useful information can be obtained from such a chaos.
- **deep chaos**: Shown in Figure 5.14(b), this dynamics has more structure in its power spectrum but it is still very close to white noise. Still, broad peaks and their harmonics are clearly distinguishable. The return map however shows no difference from the previous case.
- **informative chaos**: Shown in Figure 5.14(c), this dynamics shows a structured power spectrum where nearby limit cycles show up as a peak.

[7] As defined in section 6.2.5 on page 51



(a) Road to chaos when learning 3 cycles of size 5. N=25



(b) Road to chaos when learning 2 cycles of size 4. N=9

FIGURE 5.13. Roads to chaos through Hebbian learning in recurrent neural networks. These plots compute the bifurcation diagrams, with the learning time step playing the role of control parameter. One neuron is chosen randomly on the network, and at each learning step, its orbit is plotted.

The other periods indicate unpredictable chaotic itinerancy. The return map is very structured and sets itself apart from the two previous cases. In fact, by slightly modifying the external stimuli, the network falls in a nearby limit cycle attractor.

Before giving a more formal definition, it is interesting to note that the white noise and deep chaos are very similar and their distinction does not yield much interest. It is easy to observe the difference in terms of power spectrum analyses but when looking at their trajectory, it seems much harder to precisely define their difference. This thesis take a particular interest in the third type of chaos, which clearly shows a lot of differences and thus two definitions are provided. One defining a general chaotic attractor, and one defining the frustrated chaotic attractor which is a particular case of the chaotic attractor (defined in Section 2.2.3 on page 120).

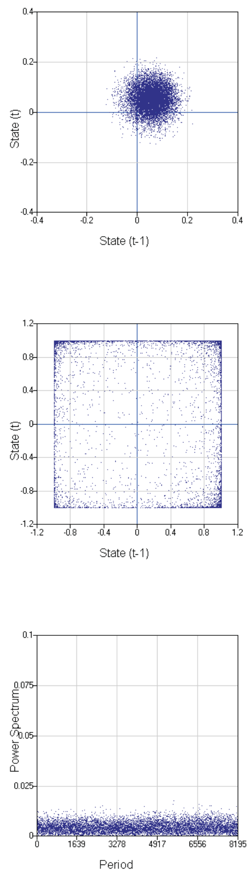
It is not an easy task to predict which kind of dynamics will be present in the system but some guidelines can be found. For instance, learning one cycle of increasing length brings deep chaos at first and ends up with white noise after some point. A small learning set composed of cycles generally brings informative chaos, where there is a competition amongst nearby attractors (which is the source of the chaotic behaviors). Again, when the data set becomes larger, the dynamics looses its structure and tends to behave as a white noise. All information about hypothetic nearby limit cycle attractors is lost. Chaotic dynamics observed in surrogate networks is generally white noise. This is probably the reason behind the large mean Lyapunov exponent.

ε_s	0.05	It is chosen quite small in order to avoid brutal changes. From experimentation, a smaller value slows the learning process too much, and a larger value induces too strong changes.
ε_b	0.05	It is kept similar to the inner state's learning rate.
ln_s	0	The learning noise is not needed for these experiments.
ln_b	0	The learning noise is not needed for these experiments.
l_μ	$*$	These experiments deal with various size cyclic attractors.
k	10	Used to compute the size of the transient phase to skip.
$\lvert A\rvert$	100	Size of the associative layer.
$\lvert S\rvert$	100	Size of the input layer.

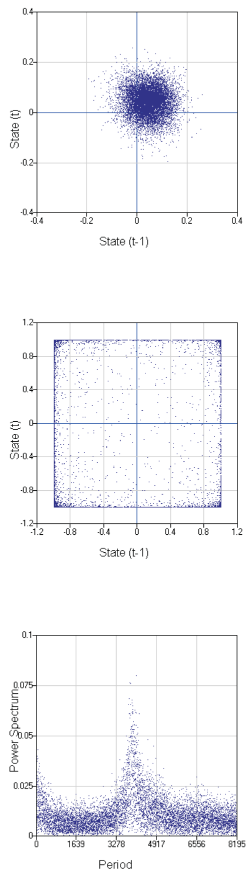
TABLE 8. This table recaps all the parameters used for the experiments in Section 2.1.3 on page 115 with their value. A value shown as $*$ indicates that this variable takes different values during the experiment, and those values are reported on the corresponding figure.

2.2. "In-supervised" learning.

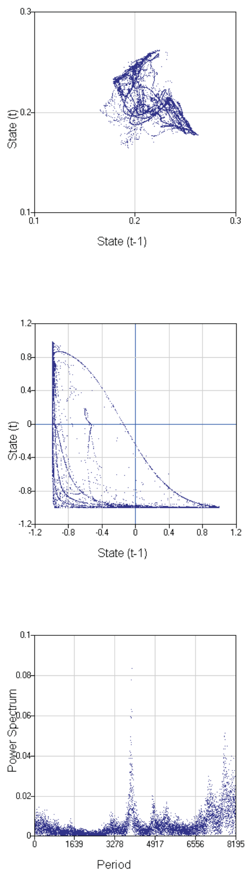
2.2.1. *Spontaneous regime.* The experiments performed with the "out-supervised" algorithm are repeated here for the "in-supervised" counterpart and the outcome is very different from the results obtained with the first algorithm. Figure 5.15 on page 119 shows the mean Lyapunov exponent and the probability of chaotic dynamics for networks learning an increasing size data set of period-2 cycles. The "in-supervised" algorithm shows larger standard deviations for certain size. This comes from the same reason as for the "out-supervised" algorithm, since this algorithm can also leads sometimes to stable systems which may be the source of the irregularities of the standard deviations.



(a) Chaotic regime obtained in a constrained random network. The lack of structure makes it very similar to white noise.



(b) Chaotic regime obtained in a network which has learned 1 cycle of size 35. Deep chaos showing less structure can be observed.



(c) Chaotic regime obtained in a network which has learned 5 cycles of size 10. The different peaks show the presence of nearby limit cycles among which the network hesitates.

FIGURE 5.14. The three most characteristic chaos observed after learning. These examples are observed as spontaneous regime in the network after learning. Top row shows the return map of the mean signal, middle row shows the return map of a single neuron and bottom row exhibits the power spectrum. The size of the layers is set to $|A| = |S| = 100$.

The main difference between this result and the one shown in Figure 5.11 on page 114 is that the presence of chaos is bounded for the "in-supervised" algorithm. Even after extensive training, the probability to have chaos does not increase beyond a certain point, while with the "out-supervised" algorithm this probability tends to one very quickly. This is explained by the fact that this algorithm is based on a process of trials, errors and adaptations that provides robustness and prevents full chaos. By increasing the data set to learn, learning takes more and more time, but at the same time, the number of readjustments increases and forces large basins of stable dynamics. Through learning, the network is constrained, and complexity increases, but does not become fully chaotic.

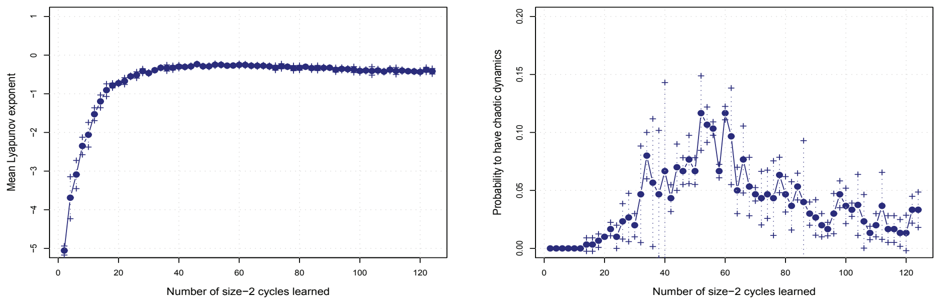


FIGURE 5.15. Dynamics of the network after an "in-supervised" learning. Shown as the mean and standard deviation of the dynamical properties of 1000 networks after "in-supervised" learning. The data set is composed of period-2 cycles. Chaos arises with learning but, unlike "out-supervised" learning, this time it is bounded (mean Lyapunov seems to have a threshold around 0) and does not pollute the whole state space (the probability of chaos never rises above 15%). The size of the layers is set to $|A| = |S| = 25$.

ε_s	0.05	It is chosen quite small in order to avoid brutal changes. From experimentation, a smaller value slows the learning process too much, and a larger value induces too strong changes.
ε_b	0.05	It is kept similar to the inner state's learning rate.
ln_s	0	The learning noise is not needed for the "in-supervised" learning
ln_b	0	The learning noise is not needed for the "in-supervised" learning
l_μ	2	These experiments deal with size-2 cyclic attractors.
$\mathtt{min}_{cs}$	2	Minimum and maximum cycle sizes are set equals to be fair to the "out-supervised" learning in the comparison.
$\mathtt{max}_{cs}$	2	Minimum and maximum cycle sizes are set equals to be fair to the "out-supervised" learning in the comparison.
p_s	2	Preferred compression period must be in $[\mathtt{min}_{cs}, \mathtt{max}_{cs}]$.
p_e	2	Preferred expansion period must be in $[\mathtt{min}_{cs}, \mathtt{max}_{cs}]$.
κ_i	3	Size of the input layer.
k	10	Used to compute the size of the transient phase to skip.
$\|A\|$	25	Size of the associative layer.
$\|S\|$	25	Size of the input layer.

TABLE 9. This table recaps all the parameters used for the experiments in Section 2.2.1 on page 117 with their value.

2.2.2. *Nature of chaos.* The next step is to give a qualitative description of the chaotic regime encountered in these networks; this is done again with the return maps and power spectrum analyses.

The previous section showed three types of chaotic regimes appearing in networks learned through the "out-supervised" algorithm. Unfortunately, when the size of the data set increases only two of those original three types of chaos were to be found. The informative chaos seems to vanish in favor of deep chaos or white noise, which clearly indicates that the networks become unstructured and this is also reflected in the content-addressability of the system which vanishes after some point. The result in Figure 5.9 on page 111 is easier to understand in that regard: even a slight modification of the input leads the system from the limit cycle attractor's weak basin into a very strong chaotic one.

On the other hand, as suggested by the Figure 5.16 on the facing page, the chaos present in a network after "in-supervised" learning is more informative. This figure shows the network in an initial stable condition then through four successive modifications of the external stimulus: the network is slowly shifted to another stable original output. The chaotic dynamics observed in these plots shows the influence of the nearby limit cycles[8] (including the one the network has started in and the one it ends up in).

When learning by means of the "in-supervised" procedure, chaotic dynamics are nearly all the time of the third type: very informative chaos shows up. By not predefining the internal representations of the network, this type of learning preserves more structure in the chaotic dynamics.

Figure 5.17 on page 122 compares Lyapunov spectra obtained from "deep chaos" in "out-supervised" learned networks and from "informative chaos" in "in-supervised" learned networks. From this figure, "deep chaos" can be related to "hyper-chaos" [Rössler, 1983]. In hyper-chaos, the presence of more than one positive Lyapunov exponent is expected and these exponents are expected to be high. By contrast, Lyapunov spectra obtained in "informative chaos" after "in-supervised" learning are characteristic of chaotic itinerancies [Kaneko and Tsuda, 2003]. In this type of chaos, the dynamics is attracted to learned memories –which is indicated by negative Lyapunov exponents– while in the same time it escapes from them –which is indicated by the presence of at least one positive Lyapunov exponent. However, this positive Lyapunov exponent must be slightly positive in order not to erase completely the system's past history and thus to keep traces of the learned memories. This regime of frustration is increased by some modes of neutral stability indicated by the presence of many exponents whose values are close to zero.

2.2.3. *The frustrated chaos.* About two decades ago, chaotic itinerancy was proposed as a universal dynamical concept in high-dimensional systems [Ikeda et al., 1989; Kaneko, 1992; Kaneko and Tsuda, 2003; Tsuda, 1992]:

> Chaotic itinerancy can be described by an itinerant motion among varieties of ordered states through high-dimensional chaotic motion.

This type of chaos has been observed in neural networks on many occasions; Bersini has shown that, for some specific weight configurations, it was possible to drive small recurrent neural networks in similar very informative and structured chaotic dynamics [Bersini, 1998; Bersini and Calenbuhr, 1997; Bersini and Sener, 2002]. He has identified this as originating from a physical frustration phenomenon. Therefore, it has been named "frustrated chaos":

> Frustrated chaos is a dynamical regime which appears in a network when the global structure is such that local connectivity patterns responsible for stable oscillatory behaviors are intertwined, leading to mutually competing attractors and unpredictable itinerancy among brief appearance of these attractors.

[8]This chaos has been described by Bersini [1998] as a frustrated chaos. The frustrated chaos is covered in more depth in the next section.

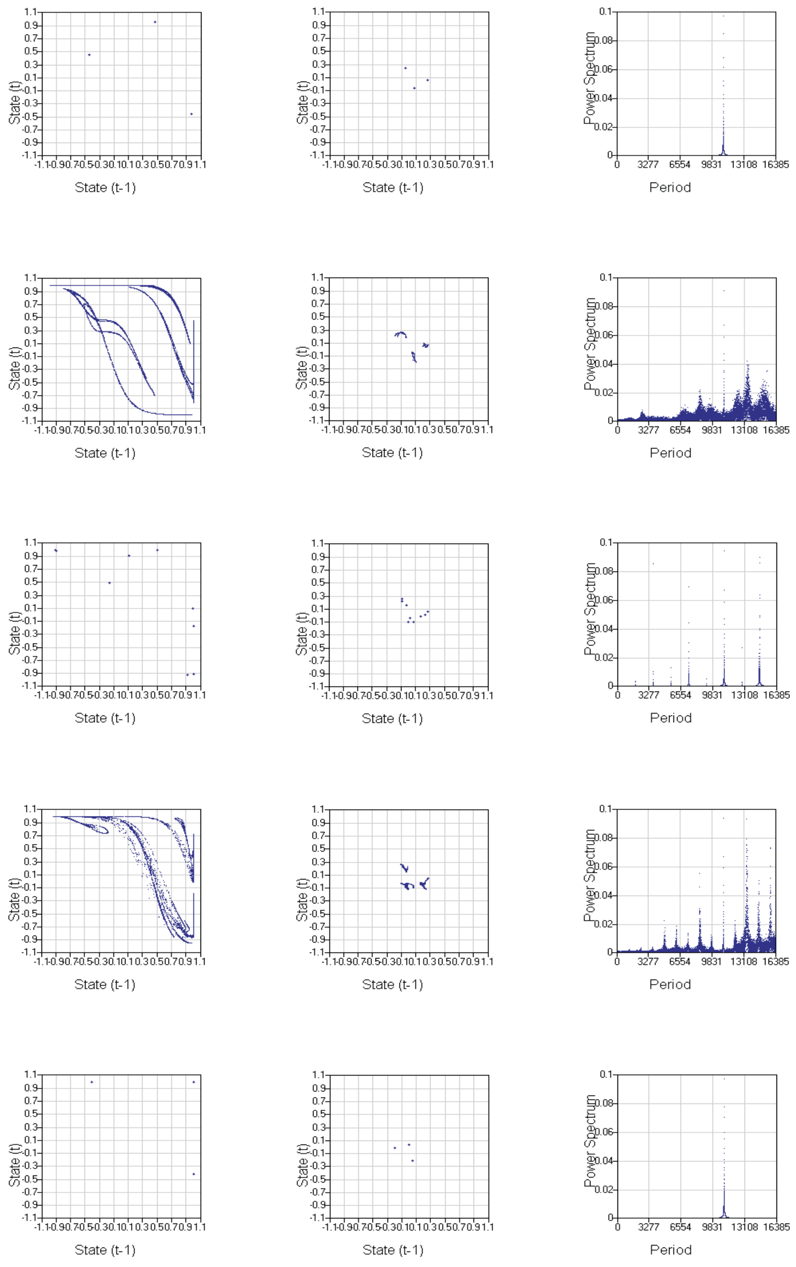


FIGURE 5.16. Variation of the network's dynamics. Through return maps of one neuron (left), return maps of the mean signal (middle) and power spectra of the mean signal (right). The external stimulus is slowly modified from one learned stimuli to another one (five steps are shown here, including the first and last ones). The 25 neurons network has learned 15 stimuli in period-4 cycles with the "in-supervised" learning algorithm.

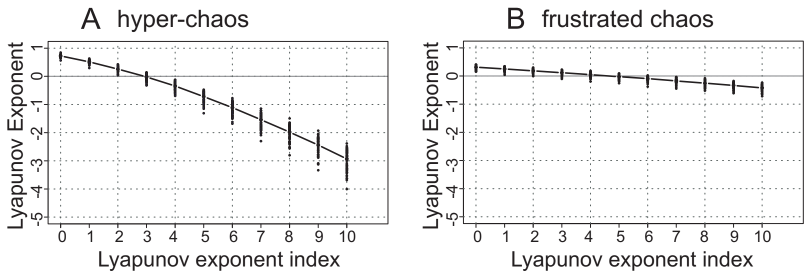


FIGURE 5.17. Lyapunov spectrum of the two learning algorithms. The first ten Lyapunov exponents are computed: (A) after "out-supervised" learning of 10 period-4 cycles, (B) after "in-supervised" learning of 20 period-4 cycles. Each time, 10 different learning sets have been tested. For each obtained network, 100 chaotic dynamics have been studied. All obtained Lyapunov spectra are plotted to show the variations.

ε_s	0.05	It is chosen quite small in order to avoid brutal changes. From experimentation, a smaller value slows the learning process too much, and a larger value induces too strong changes.
ε_b	0.05	It is kept similar to the inner state's learning rate.
ln_s	0	The learning noise is not needed for the "in-supervised" learning
ln_b	0	The learning noise is not needed for the "in-supervised" learning
l_μ	4	These experiments deal with size-4 cyclic attractors.
$\mathtt{min}_{cs}$	4	Minimum and maximum cycle sizes are set equals to be fair to the "out-supervised" learning in the comparison.
$\mathtt{max}_{cs}$	4	Minimum and maximum cycle sizes are set equals to be fair to the "out-supervised" learning in the comparison.
p_s	4	Preferred compression period must be in $[\mathtt{min}_{cs}, \mathtt{max}_{cs}]$.
p_e	4	Preferred expansion period must be in $[\mathtt{min}_{cs}, \mathtt{max}_{cs}]$.
κ_i	3	Size of the input layer.
k	10	Used to compute the size of the transient phase to skip.
$\lvert A\rvert$	25	Size of the associative layer.
$\lvert S\rvert$	25	Size of the input layer.

TABLE 10. This table recaps all the parameters used for the experiments in Section 2.2.2 on page 118 with their value.

Both descriptions of chaos fit under the more general intermittency chaos [Pomeau and Manneville, 1980] type. They are both characterized by strong cyclic components among which the dynamics randomly itinerates; it is in the transparency and the exploitation of those cycles that lies the key difference. In the frustrated chaos however, those cycles are the basic stages of the road to chaos. It is by forcing those cycles in the network (by tuning the connection parameters) that the chaos finally shows up, while the chaotic itinerancy is merely observed on random networks. One way of forcing these cycles is indeed by the time asymmetric Hebbian learning proposed in the previous chapter.

Formally defining a frustrated chaos is beyond the scope of this work; yet, to be able to work with this concept an operational definition is provided.

Definition 8. *A* symbolic attractor *is an attractor obtained by quantizing the output of the system into symbols and looking at those sequences of symbols.*

For example, if the output is quantized into 8 letters from 'a' to 'h', then one possible symbolic attractor would be 'agce'.

Definition 9. *A chaotic attractor of a system is said to be* frustrated *if its symbolic output contains recurring patterns of some of the learned non-chaotic symbolic attractors.*

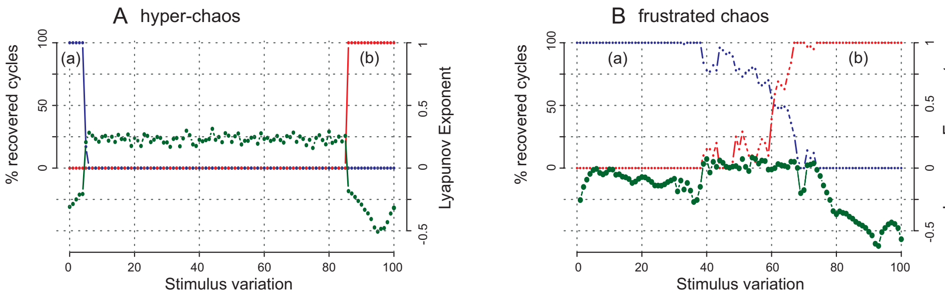


FIGURE 5.18. Presence of the nearby limit cycle attractors. Computed as the probability to find nearby limit cycle attractors in a chaotic dynamics (y-axis). By slowly shifting the external stimulus from a stimulus previously learned (region (a)) to another learned stimulus (region (b)), the network's dynamics goes from the limit cycle attractor associated to the former stimulus to the limit cycle attractor associated to the latter stimulus. The probabilities of presence of cycle (a) and of cycle (b) are plotted (in blue and red respectively). The left plot shows that, after "out-supervised" learning, once the external stimulus is outside the noise tolerance of the system, any trace of the attractor is lost. The right plot shows that, after "in-supervised" learning, when an external stimulus is too ambiguous, there are still nearby attractors. The green curves show the respective Lyapunov exponent; as expected, when outside the basins of attraction, the network learned with the "out-supervised" algorithm shows signs of a hyper-chaos, while the network learned with "in-supervised" shows a frustrated chaos. Incidentally, this plot also confirms the stronger basin of attraction with the "in-supervised" learning.

To have a better understanding of this type of chaos, Figure 5.18 compares an hyper-chaos and the frustrated one through the probability of presence of the nearby limit cycle attractors in chaotic dynamics. In the figure on the left, the network has learned two data in limit cycles attractors of period 10 by using the "out-supervised" Hebbian algorithm. This algorithm hardly constrains the network and, therefore, chaotic dynamics appear very uninformative: by shifting the external stimulus from one attractor to another one, in between, a strong chaos shows up (indicated by the Lyapunov exponent) where any information concerning these two limit cycle attractors is lost. In contrast, when mapping 4 stimuli to period 4 cycles, using the "in-supervised" algorithm (Figure on the right), shifting the external stimulus from one attractor to another one, the chaos encountered on the road appears much more structured. A small Lyapunov exponent and strong presence of the nearby limit cycles is easy to observe, shifting progressively from one attractor to the other one.

ε_s	0.05	It is chosen quite small in order to avoid brutal changes. From experimentation, a smaller value slows the learning process too much, and a larger value induces too strong changes.
ε_b	0.05	It is kept similar to the inner state's learning rate.
ln_s	0	The learning noise is not used in order to have smaller basin for the attractor and thus ease the study of the in between dynamics
ln_b	0	The learning noise is not used in order to have smaller basin for the attractor and thus ease the study of the in between dynamics
l_μ	4	These experiments deal with size-4 cyclic attractors.
$\texttt{min}_{cs}$	4	Minimum and maximum cycle sizes are set equals to be fair to the "out-supervised" learning in the comparison.
$\texttt{max}_{cs}$	4	Minimum and maximum cycle sizes are set equals to be fair to the "out-supervised" learning in the comparison.
p_s	4	Preferred compression period must be in $[\texttt{min}_{cs}, \texttt{max}_{cs}]$.
p_e	4	Preferred expansion period must be in $[\texttt{min}_{cs}, \texttt{max}_{cs}]$.
κ_i	3	Size of the input layer.
k	10	Used to compute the size of the transient phase to skip.
$\|A\|$	25	Size of the associative layer.
$\|S\|$	25	Size of the input layer.

TABLE 11. This table recaps all the parameters used for the experiments in Section 2.2.3 on page 120 with their value.

3. Using Complex dynamics

3.1. Symbolic analyzes. After learning, when the network is presented with a learned stimulus while its initial state is correctly chosen, the expected spatio-temporal attractor is observed [see Molter, Salihoglu, and Bersini, 2006a]. Furthermore, section 1 on page 101 has shown that noise tolerance is expected: the external stimulus or the network's internal state can be slightly modified without affecting the result. However, in some cases, the network fails to "understand" the stimulus and an unexpected symbolic attractor appears in the output. The question is to know whether this attractor should be considered as a spurious data or not.

The very intuitive idea is that, if the attractor is chaotic or if its period is different from the learned data, it is easy to recognize it at a glance, and thus to discard it. In contrast, this becomes more difficult if the observed attractor's period is the same as the ones of the attractors learned. In such a case, it is in fact impossible to know whether this information is relevant without comparing it with all the learned data. Consequently, a spurious data is defined as an attractor that has the same period as the learned data but is different from all of them.

As a result, two classification schemes are used to differentiate the symbolic attractors obtained. The first classification scheme is based on their periods. This criterion enables to distinguish between chaotic attractors, periodic attractors whose periods differ from the learned ones (named "out of range" attractors) and periodic attractors having the same period as the learned ones. The aim of the second classification scheme is to differentiate these attractors, based on the normalized Hamming distance between them and the closest learned data (i.e. the attractors at a distance less than 10%, less than 20%, etc.).

Figures 5.19 on page 126 shows the proportion of the different types of attractors found as the size of the learned data set is increased. Results obtained with the "in-supervised" and the "out-supervised" algorithm while mapping stimuli in spatio-temporal attractors of various periods are compared. For each data set size, statistics are obtained from 100 different learned networks, and each time, 1000 symbolic attractors obtained from random stimuli and internal states have been classified (the variations observed over the multiple networks were marginal). In this plot five lines show the cumulative probability to obtain different kinds of attractors. There are four labeled lines (0%,10%,20%,100%) and an unlabeled line which divides the probability space of the output as:

- **The 0% line** shows the probability to obtain a perfect recall (noted l_0),
- **The 10% line** shows the probability to obtain an attractor of correct periodicity with $d_H \leq 10$[9] (noted l_{10}),
- **The 20% line** shows the probability to obtain an attractor of correct periodicity with $d_H \leq 20$ (noted l_{20}),
- **The 100% line** shows the probability to obtain an attractor of correct periodicity with $d_H \leq 100$, i.e. any attractor of correct periodicity (noted l_{100}),
- **The unlabeled line** shows the probability to obtain any non-chaotic attractor (noted l),
- **The line** $y = 1$ shows any attractor (noted 1).

From this division, several interesting categories can be extracted:

- l_0 shows the probability to obtain a perfect recall,
- $1 - l$ shows the probability to obtain a chaotic attractor,
- $l - l_{100}$ shows the probability to obtain an ill-periodic attractor,
- $l_{100} - l_X$ shows the probability to obtain a spurious attractor when an attractor of correct periodicity with a $d_H \geq X$ is considered spurious. For example if only perfect recalls are not considered spurious then $X = 0$. Usually above 20% of the output start to lose any resemblance with the expected attractor.

When stimuli are mapped into fixed-point attractors, the proportion of chaotic attractors, and of "out of range" attractors, falls rapidly to zero. In the "out-supervised" learning algorithm this is shown by $l_{100} = l = 1$, while in the "in-supervised" learning algorithm with the increasing size of the data set this is shown by $1 - l \to 0$,$l - l_{100} \to 0$ and $l_{100} \to 1$. In contrast, the number of spurious data increases drastically. In the "out-supervised" learning algorithm this is shown by $l_{100} = 1$ and $l_{20}, l_{10}, l_0 \searrow$, while in the "in-supervised" learning algorithm l_{20} is steady,$l_{10}, l_0 \searrow$ and $l_{100} \nearrow$. In fact, both learning procedures tend to stabilize the network by enforcing symmetric weights and positive auto-connections[10].

When stimuli are mapped to cyclic attractors, both learning procedures lead to an increasing number of chaotic attractors ($l \searrow$). The more you learn, the more the spontaneous regime of the network tends to be chaotic. Still, the two learning procedures have to be differentiated. "Out-supervised" learning -due to its very constraining nature- drives the network into a fully chaotic state which prevents the network to learn more than 13 period-4 cycles ($l < 0.2$,$l_{100} < 0.1$). The less constraining and more natural "in-supervised" learning task leads to a different behavior ($l > 0.4$). This time, the network does not become fully chaotic and the storing capacity is enhanced. Unfortunately, the number of spurious data is also increasing and a noticeable proportion of spurious data is visible when the network is fed with random stimuli ($l_{100} \nearrow$).

[9]Where d_H is the normalized Hamming distance

[10]If no noise is injected during the learning procedure, the network converges to the identity matrix.

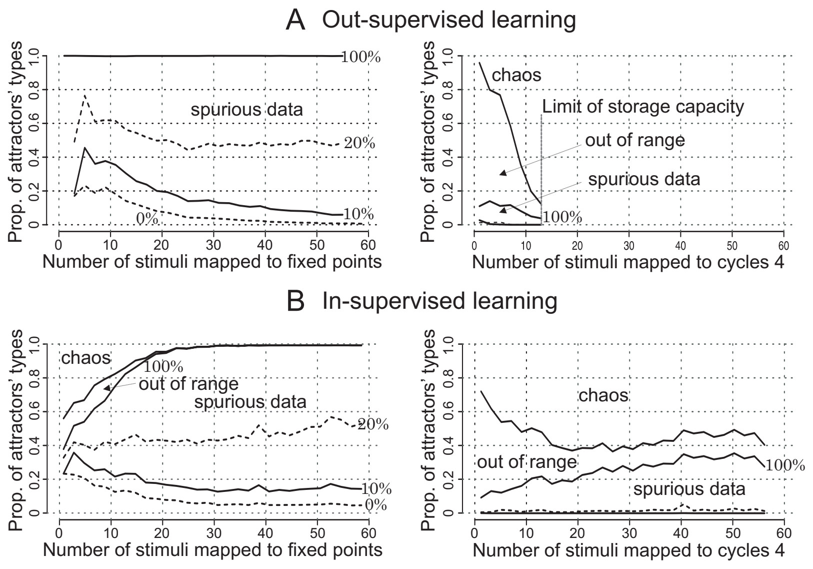


FIGURE 5.19. Proportion of the different symbolic attractors. Obtained during the spontaneous activity of artificial neural networks learned using respectively the "out-supervised" algorithm (A) and the "in-supervised" algorithm (B). Two types of mappings are analyzed: stimuli mapped to fixed-point attractors and stimuli mapped to spatio-temporal attractors of period 4. The plots shows the cumulative probability to have a perfect recall (below 0%), less than a normalized Hamming distance $d_H \leq 10$ (below 10%), $d_H \leq 20$ (below 20%), $d_H \leq 100$ (below 100%), any ill-periodic output (between 100% and the unlabeled line), chaotic output (above the unlabeled line). The spurious data are the one with a correct period but a large Hamming distance (depending on the problem, any range between 100% and 0% (or above) can be representative of this group).

The aim of Figures 5.20 on the next page is to analyze the proportion of the different types of attractors observed when the external stimulus is progressively shifted from a learned stimulus to a random one. Again, the network's initial state is set completely random. Two types of "in-supervised" mappings are compared: stimuli mapped to fixed-point attractors and stimuli mapped to attractors of period 4.

When un-noised learned stimuli are presented to the network, stunning results appear. For fixed points learned networks, in more than 80% of the observations, the data is spurious! Indeed, the distance between the observed attractors and the expected one is bigger than 10% and they have the same period (period one). By contrast, for period-4 learned networks, the probability to recover the perfect cycle increases to 56%. Moreover, 62% of the obtained cycles are at a distance less than 10% of the expected ones, and the amount of chaotic and "out of range" attractors are respectively equal to 23% and 8%. Space left for spurious data becomes less than 5%! Thus, the recall procedure can be improved by slightly modifying the network's states in case a chaotic trajectory is encountered, instead of just discarding the erroneous output. The process, repeated until a cyclic attractor is found, can usually guarantee the correct mapping. However, it is still possible

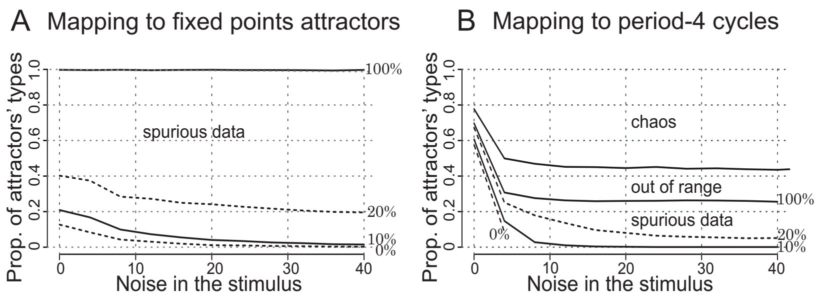


FIGURE 5.20. Proportion of the different types of attractors, observed in "in-supervised" learned networks while the noise injected in a previously learned stimulus is increased. Networks' initial states were randomly initiated. (A) 30 stimuli are mapped to fixed-point attractors. (B) 30 stimuli are mapped to period-4 cycles. 100 learned networks, with each time 1000 configurations have been tested (and the variability is negligible).

to fall into a spurious attractor or to have a chaotic attractor with a large basin which will be tolerent to those perturbations.

Because of the pervading number of spurious data in fixed-point attractors learned networks, it makes difficult to imagine these networks as working memories. By contrast, the presence of chaotic attractors helps to prevent the proliferation of spurious data in cyclic attractors learned networks while good storage capacity is possible by relying on "in-supervised" mappings.

ε_s	0.05	It is chosen quite small in order to avoid brutal changes. From experimentation, a smaller value slows the learning process too much, and a larger value induces too strong changes.
ε_b	0.05	It is kept similar to the inner state's learning rate.
ln_s	0	The learning noise is null in order to keep the test fair for the "out-supervised" learning algorithm
ln_b	0	The learning noise is null in order to keep the test fair for the "out-supervised" learning algorithm
l_μ	4	These experiments deal with size-4 cyclic attractors.
$\mathtt{min}_{cs}$	4	Minimum and maximum cycle sizes are set equals to be fair to the "out-supervised" learning in the comparison.
$\mathtt{max}_{cs}$	4	Minimum and maximum cycle sizes are set equals to be fair to the "out-supervised" learning in the comparison.
p_s	4	Preferred compression period must be in $[\mathtt{min}_{cs}, \mathtt{max}_{cs}]$.
p_e	4	Preferred expansion period must be in $[\mathtt{min}_{cs}, \mathtt{max}_{cs}]$.
κ_i	3	Size of the input layer.
$\|A\|$	25	Size of the associative layer.
k	10	Used to compute the size of the transient phase to skip.
$\|S\|$	25	Size of the input layer.

TABLE 12. This table recaps all the parameters used for the experiments in Section 3.1 on page 124 with their value.

3.2. The retrieval phase. As suggested in the previous section, the dynamics obtained outside the basins of attraction of the learned attractor is usable to further enhance content-addressability. This is even more appealing with the "in-supervised" algorithm where it clearly appears that the dynamics outside the boundaries of the known attractors shows trace of those attractors (see Figure 5.16 on page 121 and 5.18 on page 123 for more details). The idea here is to use dynamical properties of the signal as a way to identify the convergence when an external stimulus feeds the system. Given an output, any ill-periodic or chaotic attractor can be easily identified without a priori knowledge of the data set the network has learned. This allows the system to perturb those orbits in order to stabilize them in a correct attractor. This clearly helps solving one of the most common issues of associative memories, that "other" states besides the memory states may also be attractors of the dynamics. As shown in the previous section, the spurious outputs are very limited once the ill-periodic and chaotic attractor have been identified as wrong [see Salihoglu et al., 2007].

If the attractor is chaotic (or if its period is different from the learned data), it is easily recognized at a glance, and thus discarded. In contrast, things become more difficult if the observed attractor has the same period as the learned ones. In that case, it is in fact impossible to know if this information is relevant without comparing it with all the learned data. This is why only this last group is called "spurious attractor".

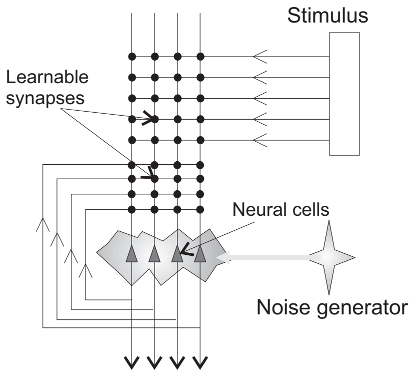


FIGURE 5.21. Noise injection mechanism. Showed here coupled with the neural network.

Here, it is showed how noise can further enhance the retrieval phase by stabilizing the chaotic trajectories to the expected ones. The following procedure is followed. First a previously learned stimulus χ^{μ} feeds the network and the network's internal state is randomly initialized. Then, after skipping the transient phase (here, arbitrarily, 24 iterations are skipped), the trajectory of the network's dynamics is observed. If a chaotic trajectory is encountered, noise is slightly applied to the network's states (by perturbing some neuron's internal state value) and the new obtained trajectory is analyzed. This new trajectory can be the desired attractor, or a spurious attractor, or again a chaotic attractor. In the latter case, the process is iterated (see Figure 5.21) again. However, there is no guarantee that this process will converge so the system must stop after a certain number of trials. Next section shows the result of this process and try to determine the optimal maximal number of iterations.

3.3. Stochastic noise and chaotic trajectories. The main result here appears on Figure 5.22 on the facing page, which shows the impact of noise during the retrieval phase. After learning, when a chaotic or ill-periodic trajectory is observed,

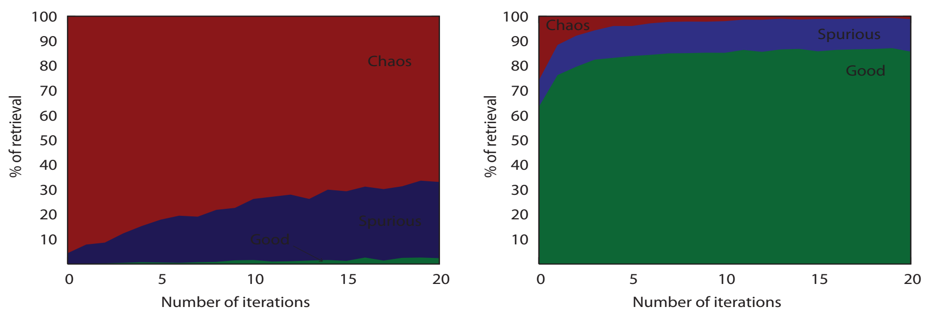


FIGURE 5.22. How noise affects memory retrieval. Left figure: after learning using the “out-supervised” algorithm, strong presence of chaos shows up. Noise does not improve the retrieval performance. Right figure: after learning using the“ in-supervised” algorithm, noise stabilizes most of the time the chaotic trajectories, and almost always into the desired attractor.

a small noise is added. This noise is uniform random noise in $[0, 0.05]$ over all the units of the associative layer. Very different results appear depending on the type of chaotic dynamics encountered. Noise cannot help that much to recover from uninformative chaotic dynamics observed after “out-supervised” learning. By contrast, noise improves consequently the retrieval performance when applied to frustrated dynamics obtained after “in-supervised” learning: performance increases from 65% to 85%, with most of the chaotic dynamics stabilized to the expected attractor.

The same figure shows that the number of iterations of the procedure of “noise stabilization” affects the result: the process has to be reiterated several times to be sure that all frustrated chaotic trajectories are stabilized. From this figure it seems that after 5 trials most of the trajectories are stabilized and from there the gain is minimal[11]. There is no real incentive to do more than 20 iterations. This is explained by the itinerating nature of chaotic trajectories. To stabilize the trajectory, this trajectory needs to be near enough the expected trajectory.

This is illustrated in Figure 5.23 on the next page, which shows the return maps of the chaotic dynamics obtained after the two kinds of learning and how noise modifies the trajectories. After “out-supervised” learning, uninformative chaos similar to white noise appears. The trajectory is not modified by noise. After “in supervised” learning, frustrated chaos appears. Noise can stabilize the chaotic trajectory to the expected attractor if it is added when the trajectory is nearby this attractor. Otherwise it may have no meaningful impact.

Figures 5.24 on the following page shows frustrated dynamics that appears when noisy stimuli are feeding the network. When such ambiguous stimuli feed the network, noise perturbation leads to jump from one chaotic attractor to another chaotic attractor. If the chaos is not too stable and nearby a stable cyclic attractor, the noise stabilizes the trajectory (like Figure 5.23 on the next page), otherwise no stabilization occurs. This figure shows this phenomenon for two different stimuli, after the initial input (red) and for each noise iteration, the return map is plotted (in order: green, blue, magenta, cyan).

[11]In terms of the number of chaotic orbits successfully stabilized into correct learned attractors

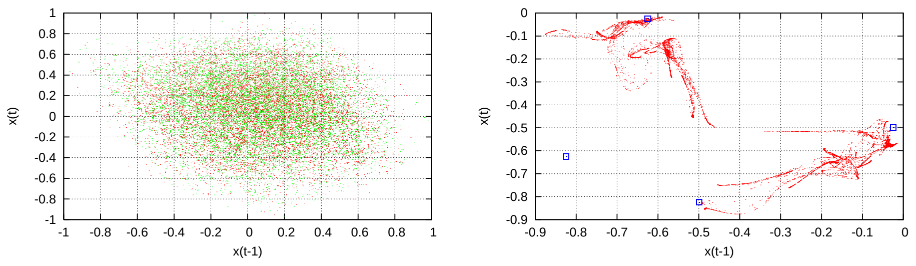


FIGURE 5.23. Return maps of the chaotic dynamics and the impact of noise observed for a learned stimulus. Left figure: uninformative chaos similar to white noise appears after "out-supervised" learning. Noise does not modify the dynamics. Right figure: very structured dynamics after "in-supervised" learning. Noise stabilizes the dynamics to a nearby cyclic attractor (highlighted with squares). Initial input's return map is shown in red, the return maps obtained by successive noise injections are shown in green, blue, magenta, cyan.

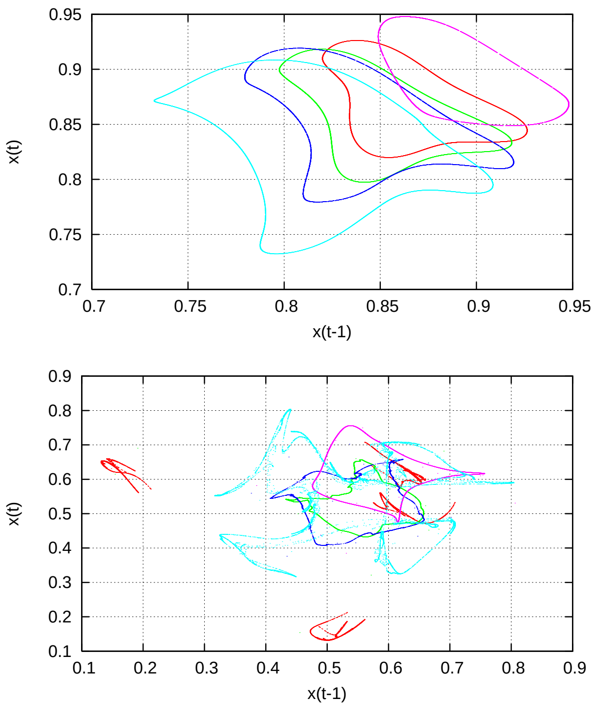


FIGURE 5.24. Return maps of the chaotic dynamics and the impact of noise when an unlearned stimulus is fed. In both examples, perturbation moves the trajectory from one chaotic attractor to another nearby chaotic attractor. Initial input's return map is shown in red, the return maps obtained by successive noise injections are shown in green, blue, magenta, cyan.

ε_s	0.05	It is chosen quite small in order to avoid brutal changes. From experimentation, a smaller value slows the learning process too much, and a larger value induces too strong changes.
ε_b	0.05	It is kept similar to the inner state's learning rate.
ln_s	0	The learning noise is null in order to keep the test fair for the "out-supervised" learning algorithm
ln_b	0	The learning noise is null in order to keep the test fair for the "out-supervised" learning algorithm
l_μ	4	These experiments deal with size-4 cyclic attractors.
$\mathtt{min}_{cs}$	4	Minimum and maximum cycle sizes are set equals to be fair to the "out-supervised" learning in the comparison.
$\mathtt{max}_{cs}$	4	Minimum and maximum cycle sizes are set equals to be fair to the "out-supervised" learning in the comparison.
p_s	4	Preferred compression period must be in $[\mathtt{min}_{cs}, \mathtt{max}_{cs}]$.
p_e	4	Preferred expansion period must be in $[\mathtt{min}_{cs}, \mathtt{max}_{cs}]$.
κ_i	3	Size of the input layer.
k	10	Used to compute the size of the transient phase to skip.
$\lvert A\rvert$	25	Size of the associative layer.
$\lvert S\rvert$	25	Size of the input layer.

TABLE 13. This table recaps all the parameters used for the experiments in Section 3.3 on page 128 with their value.

4. Conclusion

This chapter has taken the two learning procedures from the previous chapter and explored their potential. Results are in line with those from Molter and Bersini [2003a,b] regarding the usefulness of the external stimulus. Modifying the stimuli mainly results in a change of the entire internal dynamics, leading to an enlargement of the set of attractors and potential "memory bags". In particular, here this addition also increases the storing capacity and leads to an improvement of the robustness to noise.

When the system possesses external stimuli, it becomes a sort of input/output machine, where the external stimuli plays the role of input and the inner states of the recurrent neural network are the output. Even if in surface this system seems similar to the classical feed forward neural network, two major differences make the comparison not very useful. First, in the feed forward neural network output is solely dependent on the input, where here the inner states of the network also influence the outcome and thus the same input can lead to different attractors depending on the initial condition. The second difference lies in the ability of recurrent neural networks to store cyclic attractors while feed forward networks are always stable by definition. As an added consequence of these two major differences, recurrent networks can also learn different sequences sharing common patterns.

In addition to this, quantitative results have been shown describing the performance of the "out-supervised" algorithm in different memory tasks (i.e. auto-associative, hetero-associative, learning sequence), and also highlight the importance of correctly setting a learning noise that will explicitly "excavate" the basins of attraction of the stored data in order to increase the robustness. Furthermore, this learning noise has been shown mandatory during fixed-point learnings in order to exclude the trivial solution of the identity matrix, which can learn any data set but has no noise tolerance whatsoever.

The “in-supervised” algorithm has gone through similar tests and has been compared with its counterpart. Several observations were made. First the “in-supervised” algorithm is much faster than the “out-supervised” algorithm (in terms of the number of weight modifications needed to learn a given data set). It was also obvious how this algorithm can learn much larger data sets. For these tests to be fair, both algorithms were set to work with the same constraints. First, no learning noise was used ($ln = ln_b = 0$) and secondly, the “in-supervised” algorithm can only propose cycles of the same size as the one in the data set the “out-supervised” algorithm has to learn ($pe = ps = \mathtt{max}_{cs} = \mathtt{min}_{cs} = K$, with K a constant). However, it has been shown that the “in-supervised “ algorithm does not need learning noise in order to learn strong robust attractors. Furthermore, even when comparing “out-supervised” learning results with learning noise to “in-supervised” learning without noise, the latter had better content-addressability (provided by the trial–errors–adaptations process).

From these experiments it appears that the encoding capacity (with content-addressability) of “in-supervised” learned networks is greater than the maximum encoding capacities (without content-addressability) obtained from theoretical results [Gardner, 1987]. The explanation lies in the presence of the input layer, which modifies the network’s internal dynamics and enables other attractors to appear and in the same time puts this model outside the scope of those theoretical results.

The second part of this chapter takes a closer look at the different spontaneous regimes present in the system after learning different data sets. These results were compared with results obtained from random surrogate networks obtained (without learning) by replicating the weight distribution of the learned network to which it compares. It appears that global dynamics are highly related to the data set: learning static data in fixed-point attractors leads to stable networks, while learning one cycle of huge size in a limit cycle attractor entails a highly chaotic network. These networks exhibit a large set of dynamical results in which a variety of chaotic dynamics can be found. These chaotic dynamics have been identified using qualitative measures (with return maps and power spectrum) and quantitative measures (Lyapunov spectrum).

Various observations arise depending on the data set and most importantly the chosen learning algorithm. Chaotic dynamics observed in surrogate networks is most of the time similar to white noise. Networks learned using the “out-supervised” algorithm show different results. When the data set’ size comes close to the capacitive limit of the network, the observed chaos is close to white noise as well. But, with smaller data sets, chaotic dynamics show more structure, going up to a very weak and informative chaos when only a handful of short limit cycles are learned. The other important consequence of the “out-supervised” learning is observed in the probability to find chaos when looking at the spontaneous dynamics of the system. This probability tends to one, indicating that, except for the data set’s external stimuli, everything else leads to a strong uninformative chaos. This can be related to the blackout catastrophe observed in fixed point attractors networks [Amit, 1989].

Tests with the “in-supervised” algorithm show very different results. First the chaos observed in these networks is weak and very informative, also called “frustrated chaos”, since it goes from one nearby attractor to another one through short bursts. Moreover, the probability to find chaos in the network is bounded, which leaves room for more attractors to be learned and is a direct consequence of the learning process which naturally creates very robust attractors (hence with a large basin of attraction). This chaos can be very clearly observed when slowly shifting the input of the system from one known external stimulus to another one. Finally, these results are corroborated with the use of the Lyapunov spectrum which clearly shows the presence of a hyper-chaos when extensively learning with the "out-supervised" algorithm, where only a weak chaos appears after the "in-supervised" one. Coupled with the trajectory observed when shifting from one known input to another, this last chaos is clearly identified as the "frustrated chaos".

The last part of this chapter focused on symbolic analysis and how, based on the dynamical properties of these networks, the recall process can be improved. The main problem anybody faces when implementing memory-like structures is the spurious data. A spurious data is defined as an output of the system, which is not expected (based on the learning data set), yet is indistinguishable from a learned output without the explicit comparison to the whole learning data set. It has been observed that spurious data fill out the system when learning fixed point data sets, up to a point where, given an ambiguous output, it is more likely to have a spurious output than a noise recovery. Learning sequences lead to a complete different outcome: chaos fills the space and thus avoids spurious data proliferation. The "out-supervised" algorithm leads to a system where chaos takes the whole space, removing almost all spurious data. The "in-supervised" learning shows a more mitigated result since the space chaos can cover seems to have a maximum threshold (as shown with the dynamical result on the probability of chaos). However, the more interesting result is observed when the same experiment is repeated, this time giving a correct stimulus and a random inner state. It appears that in that particular setup the probability to have a spurious data is less than 5%.

This analysis is very useful; it helps improve the recall process. The main advantage of chaos taking over spurious data comes from the fact that it can be identified as a non-expected output without the need of any external information (i.e. the learned data set). The recall process has been altered to inject some small noise when facing with a chaotic output. The results were not very good for the "out-supervised" learning, the only noticeable effect being that some chaotic outputs were stabilized in some spurious data's attractor. This is due to the very uninformative nature of the chaos and thus a perturbation never guarantees to stop the system anywhere near one of the very few learned attractors, especially considering their small basins of attraction. The "in-supervised" learning algorithm, on the other hand, showed very good results and was able to increase its recovery rate from 60% to 80%.

In conclusion, this chapter discussed original results obtained [Molter, Salihoglu, and Bersini, 2007b] with the two learning algorithms introduced in the previous chapter. These results are new and provide evidence that the learning algorithms proposed here are robust and built on strong hypotheses. The capacitive limits of these algorithms is way beyond classical Hopfield networks and additionally they have been shown to be a route to chaos. Moreover, these dynamical properties have been analyzed and shown not to be just some minor side effect, since they clearly reduce the spurious data present in the system [Molter, Salihoglu, and Bersini, 2007a]. This allowed the introduction of a improved recall process which, by injecting noise on the system, allows to exploit those dynamics and improve the recalls [Salihoglu et al., 2007].

CHAPTER 6

Working Memory Model

The algorithms presented and analyzed in the two previous chapters show some promising results; yet they fail to be convincing by their lack of biological plausibility. Moreover, they are, by nature, hard to scale to larger networks since each iteration requires the modification of all the weights of the system, which rises quadratically with the size of the network. Even if those algorithms rely on a local learning rule (the iterative Hebbian algorithm), they learn the patterns globally. The iterative Hebbian algorithm does not provide any mechanism to indicate that some neuron can have any output. The sequence always specifies the global behavior of the whole system. In this chapter, a different memory system is proposed where patterns result from local connections.

The basis of this work came from the research done on the working memory models [Durstewitz et al., 2000]. These models are used for temporarily storing and manipulating information in short-term memory [Durstewitz et al., 2000]. As with any memory model, the information that is stored needs to be specified. To this end, most of the time, these models rely on cell assemblies (as described sixty years ago by Hebb [1949]).

Cell assemblies are good candidates since they use only a subset of the neurons to represent the information, which provides locality. Moreover, they can have overlaps between them, which helps improving the capacity of the system and, more importantly, a cell assembly being characterized by the synchronous activity of a group of neurons, they leave the exact behavior of these neurons unspecified and not directly related to the nature of the information. As shown with the "in-supervised" learning algorithm, letting the exact output of the system unspecified a priori helps the system in terms of capacity and content-addressability.

Working memory models having already been covered from a psychological point of view, the next section will look at those models from a computational neuroscience point of view. The second section covers the cell assembly and gives a proper definition for the concept. The last section shows how these concepts translate into the model introduced earlier (Chapter 3 on page 73).

1. Definitions

The brain is not only defined by its ability to passively store memories but also by the way it manipulates this information. This is why the memory is addressed as working memory and not just short-term memory. The working memory models deal with those short-term memories but also with the way information is manipulated during short time periods [Baddeley, 1986].

Several studies showed, using single cell recording, that during matching tasks with delay[1], primates had increased firing rates in specific areas of the brain [Fuster,

[1]In these tasks, the primate has to retain a specific information during a short period of time to guide a forthcoming response.

1973; Fuster and Alexander, 1971; Rainer et al., 1998]. Neuroimaging tools allowed Cohen et al. [1997] to observe increased activity in some region of the brain. According to this, the prefrontal cortex, the posterior parietal cortex and parts of the basal ganglia and of the thalamus are believed to be involved in the working memory [Goldman-Rakic, 1994].

In this chapter, a particular focus is given to the first defining feature of the working memory: the ability to actively hold a limited amount of information in memory for a short time, which is also called the short-term memory[2]. Many biological evidences suggest that the short-term memory is an intrinsic feature of the prefrontal cortex [Miller et al., 1996], which means that the manipulation of the sustained information (the second defining feature of the working memory) would result from the interplay between different brain areas.

All the observations about the neural basis of the working memory have inspired many neurobiologically-based computational models [Ashby et al., 2005; Durstewitz et al., 2000]. Yet, most of them do not use the cell assemblies as the substrate for information. Here, those cell assemblies are defined first according to their synaptic weights and later autonomously by the system with an unsupervised learning algorithm. The dynamics of these cell assemblies is investigated and they are used to reproduce the classical paradigm of associative memory, i.e. that the partial stimulation of a cell assembly entails its entire activation.

In recent years, different groups have reported that, in absence of any tasks or stimuli, the resting brain is not silent, but spontaneously active, and that this spontaneous activity exhibits exquisite spatiotemporal patterns of activity [Kenet et al., 2003; MacLean et al., 2005; Vincent and Buckner, 2007]. According to their findings, during spontaneous activity, the brain activity itinerates through previously learned 'memories' or 'thoughts', in a way similar to chaotic itinerancy [Tsuda, 2001].

2. Cell Assemblies

The "cell assembly theory", also called the "Hebbian theory", has already been mentioned at several occasions in this thesis. This section takes a closer look at this theory and properly defines the concepts behind it. The main idea behind this theory has already been explained and is the so called "Hebbian rule". This rule directly dictates a definition for the cell assembly concept. Hebb [1949] describes this with the following words:

> Let us assume that the persistence or repetition of a reverberatory activity (or "trace") tends to induce lasting cellular changes that add to its stability. When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.

This is often simplified as "cells that fire together, wire together". Even if this is an oversimplification, it still shows an important consequence of such plasticity: the neurons group themselves into meaningful clusters.

From this, cell assemblies can be defined as a group of cells having strong synaptic connections. This definition is based on the structural nature of the network.

[2]Not to confuse with the short-term memory as described earlier, which is present in the hippocampus and can last several months.

It gives a static view of the cell assemblies in the system. The problem with this definition is that it is a non-trivial task to identify such a cluster when working with nonlinear systems. The large number of neurons does not help either, since it becomes less likely that all neurons of a same cluster will have strong connections between each other. Finally, the input also affects the outcome of the cluster: for instance, if three neurons are strongly connected to each other and that a given input excites two of them but strongly inhibits the last one, there is a good chance that only two of the neurons will be activated. This still forms a cluster of densely connected neurons but yet all densely connected neurons do not belong to it.

To summarize these observations, even if a synchronous activity leads to strong connections, it is not safe to assume that strong connections leads to synchrony nor that all parts of a cluster will have strong connections with all the other parts. This makes it hard to define a cell assembly purely based on the static weight distribution. However, it is still possible to define them based on a dynamical specification: a group of neurons that have a synchronous activity as a response to a given input form a cell assembly. Of course, this definition is straightforward and works very well to identify the cell assembly, but it has its own drawbacks. For instance, it is impossible to know for sure if all the cell assemblies of the system have been identified since it is always possible that another unknown cluster will form when facing a previously unseen external stimulus.

Hopfield [1982] is amongst the precursors to have put this Hebbian rule to good use (see Section 5.1 on page 35 for more details). However, his model had its problems, including from a cell assembly point of view. The two major criticisms that can be made on this model is that cell assemblies are static (due to fixed-point attractors) and they cannot overlap without severely crippling the memory; Hopfield has clearly showed that the capacity of such a system was very limited and can even go to catastrophic forgetting when patterns were too correlated [Amit et al., 1985; Gardner, 1987]. Finally, the cell assemblies in Hopfield's system have the same structure than the external world, which seems very unlikely.

The learning algorithms suggested in the previous chapter have managed to erase most of those problems. They learn sequences, they can deal with overlappings pretty well, and because of the external stimuli, there is a clear distinction between internal and external representation of the information[3] (the "in-supervised" version is built on this very same concept). However, these algorithms specify the dynamics of the whole system, which is in contradiction with the locality of the cell assembly: if some cells are not needed in the cell assembly, the ideal learning mechanism should not even consider them, while these algorithms force them to have a specific value (even if this value is 0).

With the rise of popularity of working memory, cell assemblies have seen a lot of interest from large groups of researchers. In their perspective, the presence of sustained internal representations after the presentation of external stimuli results from local reverberations of activity in the recurrent connections of cell assemblies. Many researchers used cell assemblies but without putting any meaning for them: the cell assemblies are a priori defined from bi-modal weights distributions [Amit, 1995; Brunel and Wang, 2001; Molter et al., 2009; Mongillo et al., 2008]

[3]This is of course not true when learning auto-associative fixed-point data sets.

Recent improvements of recording techniques enabled the simultaneous recording of a large number of cells, which has led to numerous new analyses over population codings and spatio-temporal patterns. This has in turn increased the popularity of the cell assembly theory. For example, Harris [2005] argued that many observations that have been interpreted as evidence for temporal coding might instead reflect an underlying assembly structure. Pastalkova et al. [2008] reported that reliably and continually changing cell assemblies in the rat hippocampus appeared not only during spatial navigation but also in the absence of changing environmental or body-derived inputs. All these new works strongly support the cell assembly theory.

3. Implementation

This section's goal is to measure the potential of the model described in Chapter 3 on page 73 to exhibit working memory features. The first step in this direction is to study the system without any learning algorithm and to find out if it can exhibit coherent behaviors when put in a configuration reminiscent of biological evidences. A priori encoding cell assemblies in the synaptic weights of the recurrent neural network is tested first. This procedure follows a long history of working memory models in cell assemblies [e.g. Compte et al., 2000; Molter et al., 2008; Mongillo et al., 2008].

3.1. A priori encoding of cell assemblies. Following its first definition [Hebb, 1949], a cell assembly (CA for short) is defined by subset of cells connected between each others with high synaptic weights, compared to other synaptic weights. Accordingly, to account for the presence of cell assemblies, synaptic weights were chosen from the following uniform random distributions.

First, units (or cells) in a same cell assembly were strongly connected using a uniform random distribution ($w_{ij}^R \in [-0.95, 0.95]$). This distribution has been chosen to take into account that cell assemblies have strong synaptic connectivity amongst their cells but also that each cell in the CA does not need to be strongly connected to each others (as explained in the previous section). The reason for this distribution to include also negative weights lies in the activation function of neurons. Since this is a rate firing model, neuron outputs are always positive. So if a positive distribution was considered ($w_{ij}^R \in [0, 0.95]$) then the system is basically limited to two outputs. First, if the inhibition (from the global inhibitor) is not strong enough, then the system will end up saturating at 1; and if the inhibition is strong enough, the system will oscillate between near maximum value and near zero value. This happens because if the system gets too active, the inhibitor dampens it; then since the system has become quiet the inhibition stops, which allows the system to become saturated again. Basically, this configuration leads the system's output to be dictated by the inhibitor, which is not its the intended use.

Secondly, units which were outside cell assemblies were weakly connected together using a uniform random distribution ($w_{ij}^R \in [-0.05, 0.05]$). This distribution is chosen to give background neurons a very light activity. Finally, to prevent the simultaneous reactivation of multiple CAs, synaptic weights between units of different cell assemblies were chosen to be slightly inhibitory using a uniform random distribution ($w_{ij}^R \in [-0.35, -0.05]$). Accordingly, the activation of a cell assembly tends to inhibit the co-activation of other cell assemblies.

The other weights of the system are initialized as follows : $\mathbf{W_A} = 1$, so all the units of the associative layer have a positive and uniform impact on the inhibitor;

$\mathbf{W_I} = -1$, so that the inhibitor has an overall negative value if the incoming input is high, and thus inhibits; and finally $\mathbf{W_S}$ is initialized with a value allowing the neuron of the stimulus to have a positive connection with the corresponding cell assembly and 0 with the rest of the units.

To define cell assemblies in a straightforward and reproducible way, the following parameters were introduced [see also Molter et al., 2008]:

- r, the number of cells composing a cell assembly.
- p, the proportion of cells in a cell assembly overlapping with other cell assemblies. $r - p \cdot r$ gives the number of cells of a CA that are unique to this CA, the other $p \cdot r$ cells are shared with some other CAs.
- q, the proportion of cells lying in two cell assemblies. $q \cdot r$ give the number of cells that two CAs can share, but these cell assemblies can still share units with other cell assemblies and the other cells may be shared with other CAs, too.

Naturally, given this definition, q must always be less or equal to p.

For this part, the context was disabled ($\mathbf{W_C} = \mathbf{W_V} = 0$) to keep the example as simple as possible. Hence, the matrix weight $\mathbf{W}$ becomes:

$$\mathbf{W} = \begin{pmatrix} \mathbf{W_R} & 0 & \mathbf{W_S} & \mathbf{W_I} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & I & 0 \\ \mathbf{W_A} & 0 & 0 & 0 \end{pmatrix} \tag{73}$$

The model is composed of the associative layer, the input layer (modeling the incoming stimuli) and the global inhibitor, as shown in the Figure 6.1.

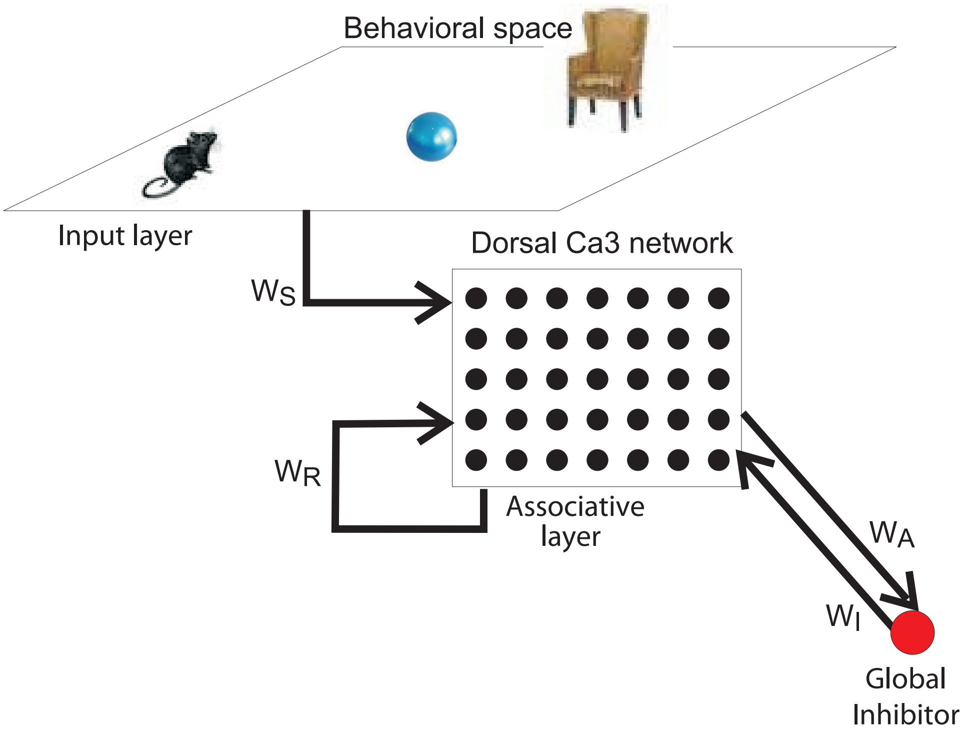


FIGURE 6.1. Limited architecture of the model, as it is used in this chapter, with only the active components.

3.2. Results. As a first experiment, the model was tested with only 2 cell assemblies pre-encoded in the network [as in Compte et al., 2000; Mongillo et al., 2008]. Each CA was composed of 30 cells ($r = 30$), with 10 overlapping cells ($p = q = 0.33\ldots$). To test this configuration, two measures are plotted. One is the average activity of all the cells in a given cell assembly, the other is a raster plot of all the neurons of the system. The raster plot shows a grid that has one row per neuron and one column per time step. The activity of a neuron i at time t is shown as the color of the corresponding cell of this grid. An inactive neuron will be represented in blue and the more active it will be, the closer to red the color will be (^{0} $x_i(t)$ ^{1}).

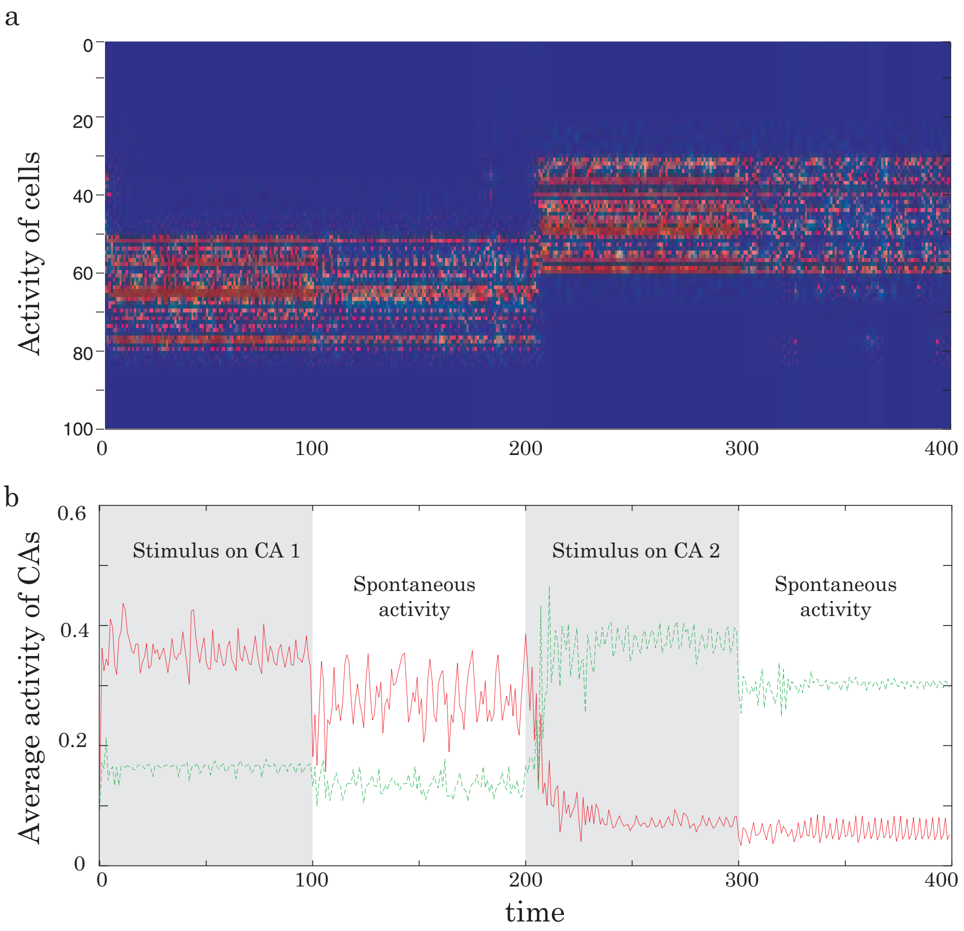


FIGURE 6.2. Content-addressable and working memory features. (a) Raster plot showing the activity through time of all units. (b) For each predefined CA, the activity through time averaged on all its units is plotted (CA 1 is red and CA 2 is green). Sequences of spontaneous activity following sequences of external stimulation impinging subsets of specific CAs were tested.

Here, the first CA is composed of the neurons 50 to 80, the second one is composed of the neurons 30 to 60, and obviously the neurons 50 to 60 are shared. Figure 6.2 shows the rasterplot activity of the cells and the averaged activity of the cell assemblies during and after an external stimulation[4] of 10% of its units (i.e. 3 cells). As expected, partial reactivation led to the reactivation of the entire CA. More importantly, the CA maintains high-level activity after the stimulation stops, which is a requirement for a working memory. In summary, this model satisfies the two requirements needed for a working memory model: CAs are correctly reactivated when fed by a partial stimulus and their activation is maintained after stimulus offset.

This work handles CA activity as an activator, and it is easy to see why. One way to see this, is to remark that a cell assembly is in fact the minimum number of cells that have a strong activity given a stimulus. This is reminiscent of the definition of an activator which has a minimality constraint. In addition, it has also a certain tolerance to noise (as shown in this section) which is reminiscent of the basin of attraction of the attractor.

The cell assembly can also be observed, as done here, purely through its activity which is a dynamical signal to which the system converges. Since the exact nature of the trajectory is not really important when looking at information through cell assembly's perspective it is acceptable that a given cell assembly can go through different dynamics. Those signals are each a different attractor of the system since the activity of the CA persists but can have any nature (cyclic, chaotic, ...). So another way to build the bridge between the CAs and the attractors of the system is to consider the set of all attractors the CA can go through as one family of attractors with a basin of attraction which is the union of all the basins of these attractors. Then again, the cell assembly fits the definition of the attractor, it is minimal since it only takes the attractor that results with the cell assembly being active. Secondly, since each basin of attraction leads to one attractor and since the CA is defined by the union of all those basins then it also has a basin of attraction that would lead to an attractor.

Differently from other working memory models where the activity of the network is sustained thanks to additional mechanisms (e.g. by a dynamical modification of the synapses [Molter et al., 2008; Mongillo et al., 2008]), in this model, the working memory feature is purely neurodynamic and results from the reverberation of the cell assembly's activity across its recurrent connections [as in Compte et al., 2000]. This figure also shows the average activity of the cell assembly itself. The dynamics of those attractors is studied later in this chapter. However, it appears clearly that they are able to generate complex behaviors.

Figure 6.3 on the following page shows the average activity of a CA when a noisy stimulus feeds the system. The system is composed of two CAs of 30 units ($p = q = 0.33\ldots$). This figure shows the activity of both CAs when the stimulus related to the first CA feeds the system. The noise in this figure is represented as a Hamming distance between the correct input and the noisy input.

In these plots, the average activity of the CA corresponding to the stimulus is always higher than the other CA until noise is high enough for the input to be close to random ($d_H = 50$). Since there is an overlap between the two CAs, it is normal for the second CA to have noticeable activity but this does not get in the way of the system to make a good recall.

Figure 6.4 on page 143 shows the same results with five cell assemblies of 10 units with weak one-to-one overlap ($q = 0.2$) and strong global overlap ($p = 0.6$). It clearly appears that, because of the overlap, all the cell assemblies show activity for

[4]The absence of external stimulus is reflected by setting the input layer to 0 ($S = 0$).

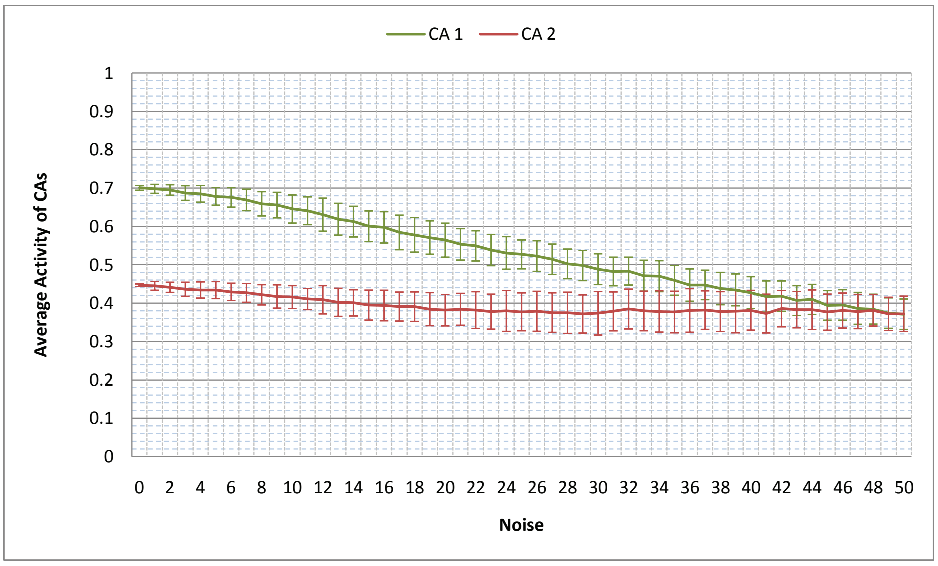


FIGURE 6.3. Noise tolerance (2 CAs). Measured as the average activity (and standard deviation) when the stimulus related to the first cell assembly feeds the system with various levels of noise. Noise is measured as a Hamming distance between the correct input and the noisy one. This test is done over 100 different random networks with two cell assemblies of 30 neurons where 10 are shared ($p = q = 0.33\ldots$).

a given stimulus. But again the correct recall is made until the stimulus becomes near random (under the effect of the injected noise).

The noise injection is performed by selecting, with a uniform random sampling without re-pick, n units in the input layer and if the unit was active as part of the stimulus its value is set to 0.0 otherwise its value is set to an active value (here, 0.3).

Those results show that a simple bi-modal weight configuration allows this model to propose a robust working memory model, since the essential features required from any such model are present: there is a substrate for information (the cell assemblies), the model is tolerant to noise (content-addressability), and the system's dynamics are strong enough to carry the information obtained by stimulating the system even after the stimulus leaves.

$\|A\|$	100	Size of the associative layer.
$\|S\|$	100	Size of the input layer.

TABLE 1. This table recaps all the parameters used for the experiments in Section 3.2 on page 140 with their value.

3.3. Dynamics analysis. The next step is to test the dynamics that can be found in these randomly generated networks and see if any specific behavior can be observed because of the particular architectural configuration required for implementing cell assemblies.

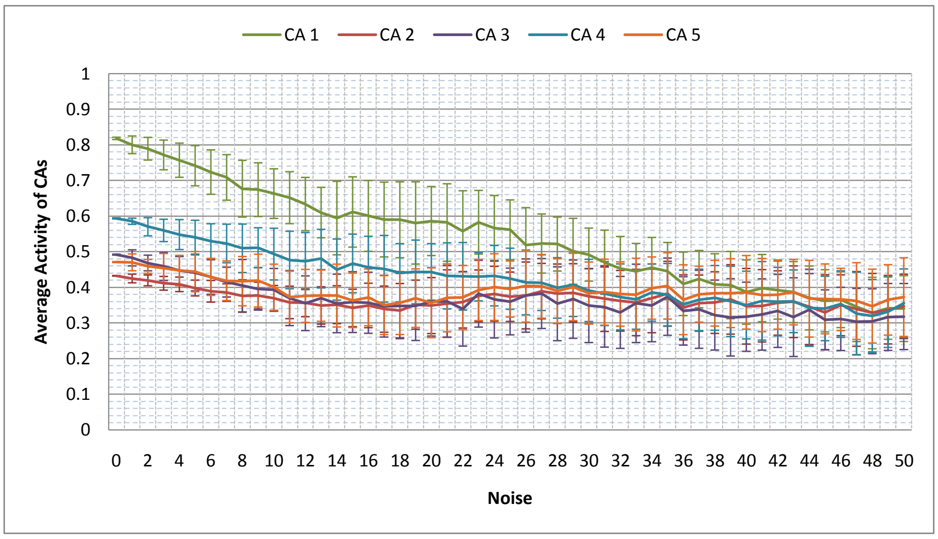


FIGURE 6.4. Noise tolerance (5 CAs). Measured as the average activity (and standard deviation) when the stimulus related to the first cell assembly feeds the system with various levels of noise. Noise is measured as a Hamming distance between the correct input and the noisy one. This test is done over 100 different random networks with five cell assemblies of 10 neurons, with weak one-to-one overlap ($q = 0.2$) and strong global overlap ($p = 0.6$).

The first tests are done on a system with two cell assemblies of different sizes and different overlappings. Figure 6.6 on page 145 shows the dynamics of a cell assembly when its corresponding stimulus feeds the system. For this experiment, two overlappings have been tested: $p = q = 0$ and $p = q = 0.5$. The size of the cell assemblies affect the dynamics they go through during simulation. But it is important to note that, depending on the overlapping factor, the influence of the size is completely different. Where non-overlapped cell assemblies configuration lead to stable output when stimulus feeds the system, the dynamics of overlapped cell assemblies are more and more chaotic when bigger cell assemblies are used.

The same experiment is repeated but this time the global dynamics of the system is observed when the stimuli feeds the system. Figure 6.6 on page 145 shows clearly that the size of the cell assembly also has an effect on the type of dynamics observed in the system as a whole. And of course, when no overlapping is present the system becomes very stable, while with overlapping it becomes more and more chaotic.

This trend is not specific to the presence of a known stimulus on the system, as shown in Figure 6.7 on page 146. The spontaneous activity of the system also shows the same correlation between size, overlapping and stability. In this figure a random stimulus feeds the system and the resulting dynamics is monitored.

A particular case can be observed when an ambiguous stimulus feeds the system. For example, when an external stimulus impinges subsets of cells belonging to two different cell assemblies (Fig. 6.8 on page 147 (top)), it is possible to find a dynamics hesitating between the two different attractors without settling down in one of them. This mechanism is likely to be derived from the chaotic nature of the attractor characterizing the cell assemblies. As observed in the previous chapter, if

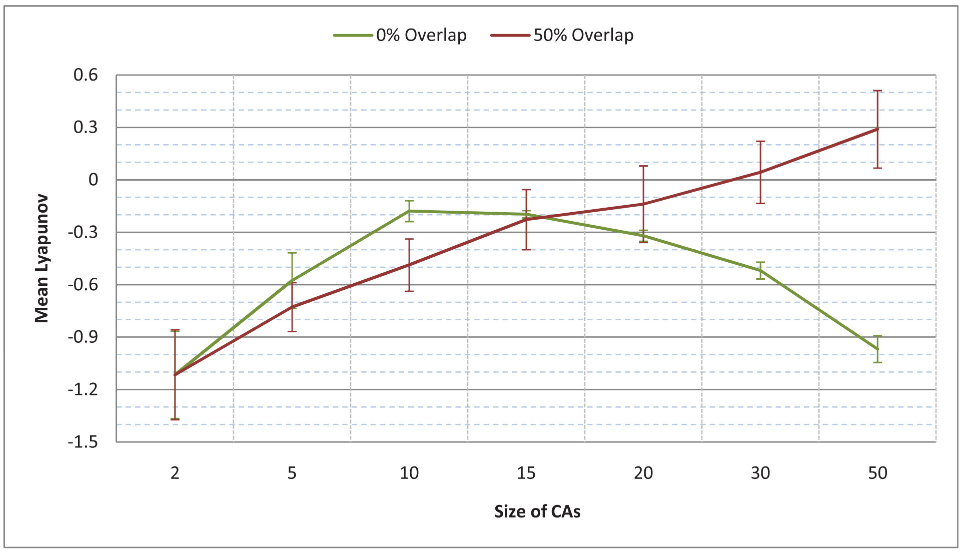


FIGURE 6.5. Dynamics of a CA during stimulus, shown as the mean Lyapunov activity of a cell assembly during stimulus. The test is done for 100 random networks with two cell assemblies. Different sizes of cell assemblies are tested with two overlapping configurations. In the first configuration (in green) the cell assemblies share no unit, while in the second configuration (in red) they share half of them. Overlapped cell assemblies show a correlation between their size and the mean Lyapunov exponent which ends up positive. Non-overlapped cell assemblies never achieve configurations with complex dynamics.

the orbit of the CA is a frustrated chaos, it will go from one attractor to another without settling. Figure 6.8 on page 147 (bottom) shows such an itinerance too: in the left part the system seems stable enough but after a certain time it brutally changes of attractor, while the figure on the right shows a competition between the two attractors with one CA activating for a short time until the other one takes over. In these examples, the input feeds the system all the time.

Other sets of experiments have been conducted to investigate the possible correlation between the number of cell assemblies and the system's dynamics. Figure 6.9 on page 148 shows how the dynamics of a stimulated cell assembly evolves with the number of cell assemblies in the system. As with the size, the number of cell assemblies reinforces the trend that was already present. In other words, when dealing with large cell assemblies the system tends to be chaotic; increasing the number of such cell assemblies make the system even more chaotic. This can be confirmed by the fact that when dealing with systems with low overlapping the number of cell assemblies stabilizes the system more and more.

Figure 6.10 on page 149 (and Figure 6.11 on page 150) leads to the same conclusions when observing the global dynamics (and spontaneous activity respectively). The plots do not have any data below 5 and above 10, for the cell assemblies of size 30 and 50: given the size of the network ($N = 100$) it was not possible to generate such a configuration.

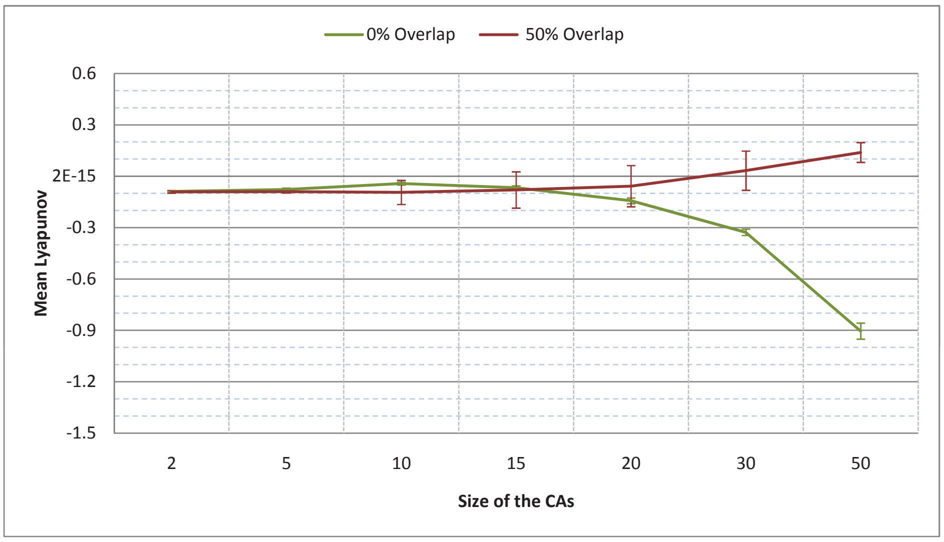


FIGURE 6.6. Dynamics of the system during stimulus. Shown as the mean Lyapunov activity of the system during stimulus. The test is done for 100 random networks with two cell assemblies. Different sizes of cell assemblies are tested with two overlapping configurations. In the first configuration (in green) the cell assemblies share no unit, while in the second configuration (in red) they share half of them. Overlapped cell assemblies show a correlation between their size and the mean Lyapunov exponent of the system, which ends up positive. Non-overlapped cell assemblies never achieve configurations with complex dynamics.

The final step was to see what part of the systems dynamics is chaotic. This is done by computing the probability to observe chaotic dynamics in the system[5]. Figure 6.12 on page 151 shows the probability of chaos for a system with two cell assemblies of various sizes and the same two overlapping configurations. Figure 6.13 on page 152 shows the same for a system with an increasing number of cell assemblies, of various sizes and overlapping configurations.

The results obtained from these tests are not very suprising. The configuration with two cell assemblies with no overlap shows no presence of chaos. However, when the same system is configured with 50% overlap chaos starts to appear in the system around cell assemblies of size 15 and nearly reaches omnipresence when using cell assemblies of size 50. From the second plot it is easy to observe that the number of cell assemblies enforces the presence of chaos and even when dealing with cell assemblies of size 30 (which show around 70% chance to have chaotic regime) this probability tends to nearly 100% when the number of such cell assemblies increases. As with the mean Lyapunov, the number of cell assemblies does not bring chaos where there was none to begin with.

[5]Which is the percentage of random inputs for which the Lyapunov exponent is greater than 0.01

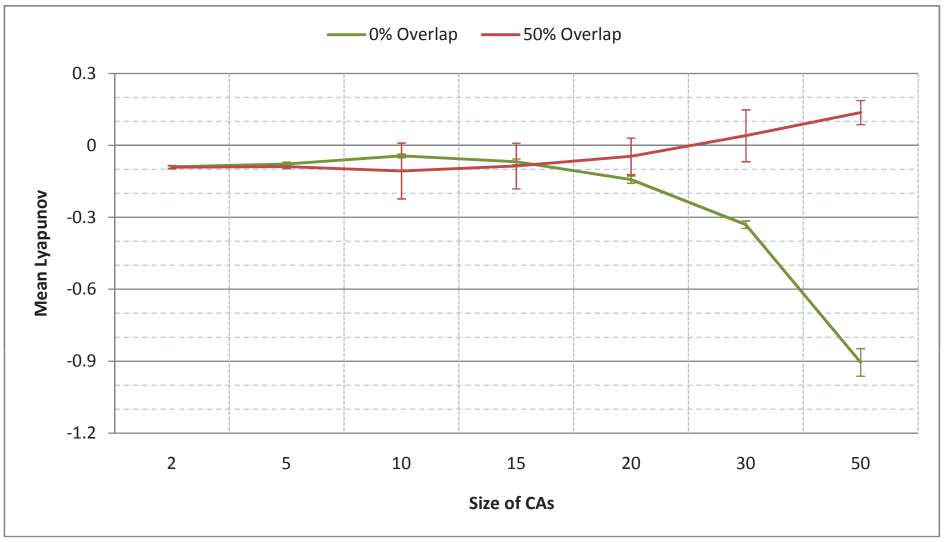


FIGURE 6.7. Spontaneous dynamics of the system. Shown as the mean Lyapunov activity of the spontaneous activity of the system. The test is done for 100 random networks with two cell assemblies. Different sizes of cell assemblies are tested with two overlapping configurations. In the first configuration (in green) the cell assemblies share no unit, while in the second configuration (in red) they share half of them. Overlapped cell assemblies show a correlation between their size and the mean Lyapunov exponent of the spontaneous activity in the system, which ends up positive. Non-overlapped cell assemblies never achieve configurations with complex dynamics.

$\|A\|$	100	Size of the associative layer.
$\|S\|$	100	Size of the input layer.

TABLE 2. This table recaps all the parameters used for the experiments in Section 3.3 on page 142 with their value.

4. Conclusion

This chapter has introduced the most known working memory models and explicits their connection with the different memory stores. All those models explicitly manipulate information chunks present in the memory , which is why it is important to define a substrate for these information.

Since 1949, the “cell assembly theory”, proposed by Hebb [1949], has gained a lot of popularity. Cell assemblies have been heavily studied over the past sixty years. Amongst other things their popularity comes from the simplicity behind their definition and the biological evidence that supports the theory. The “cell assembly theory” is closely related to the Hebbian learning rule, which is also why this theory is sometimes simply called the “Hebbian theory”. The Hebbian rule reinforces cells that activate simultaneously and weakens those that are not in sync. This leads to groups of cells to activate in concordance or in other words to form cell assemblies. Those cell assemblies have been chosen by many researchers as the substrate for encoding information.

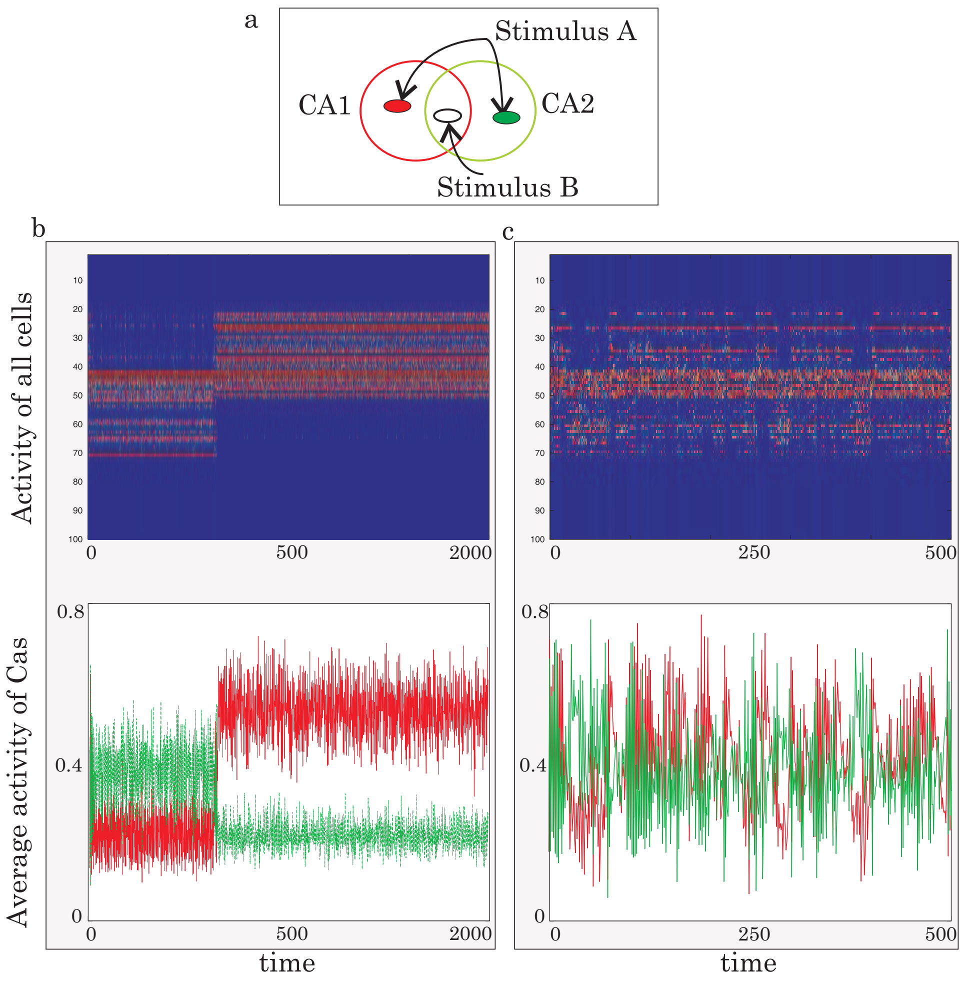


FIGURE 6.8. Network dynamics for ambiguous external stimuli. (a) An ambiguous stimulus is a stimulus which either feeds parts of two cell assemblies (Stimulus A) or part of the overlapping units (Stimulus B). (b and c) The average dynamics of each cell assembly is plotted through time. In both cases, the system exhibits complex dynamics, hesitating between the two previously stored cell assemblies' attractors. This kind of dynamics can be obtained using stimulus of type A and B.

The third part of this chapter takes the model proposed in this work and applies a priori a simple bi-modal weight configuration reminiscent of the "cell assembly theory". Once the associative layer has been defined, the model has been tested in order to determine if it was able to exhibit basic behaviors of a working memory. The first test was simply to look at the activity obtained in the network (and in particular in the predefined cell assemblies) to see if partial reactivation of a cell assembly leads to the full activation of the said cell assembly. The model shows successful result in this first test. It was also able to maintain the activity of the cell assemblies after the stimulus left the system, only using the dynamics present on the system. This is also an important feature required for working memory models. More tests have been done regarding the tolerance to noise, and even with overlapping cell assemblies, the system is very robust to noise and manages to

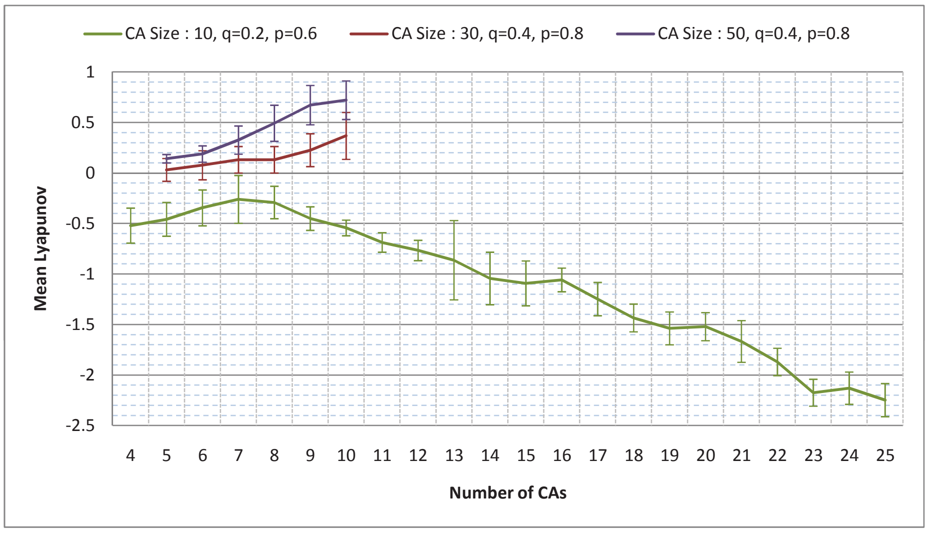


FIGURE 6.9. Dynamics of a CA during stimulus for large data sets. Shown as the mean Lyapunov activity of a cell assembly during stimulus. The test is done for 100 random networks. Different configurations of cell assemblies are tested; the number of cell assemblies present in the system is plotted on the x-axis. In the first configuration (in green) the cell assemblies have 10 units ($q = 0.2, p = 0.6$); in the second configuration (in red) they have 30 units ($q = 0.4, p = 0.8$); and finally in the third configuration (in purple) they have 50 units ($q = 0.4, p = 0.8$). The number of cell assemblies in the system enforces the type of dynamics observed when having only two cell assemblies. Smaller cell assemblies become more and more stable, while the opposite trend is observed for larger cell assemblies.

have a reactivation of the correct cell assemblies. The system has been tested with different numbers of cell assemblies with great success.

The final tests had for objective to analyze the dynamics the system present during reactivation and spontaneous activity. Different sizes and numbers of cell assemblies have been tested with various overlapping parameters. It appeared that the size of the cell assemblies and the overlapping factor affect the dynamics that can be observed in the system. The size of the cell assembly reinforces the type of dynamics the overlapping suggests, when cell assemblies have no overlapping they exhibit stable dynamics and those are more and more stable as the size of the cell assemblies increases. On the other hand, strongly overlapped cell assemblies exhibit complex dynamics which become heavily chaotic with the size of the cell assemblies. An interesting case of complex dynamics can be observed when an ambiguous stimulus feeds the system. Some ambiguous stimulus pushes the system into an unstable configuration where the system's output goes back and forth between the different cell assemblies. Interestingly, this oscillation between cell assemblies can take different forms: it can be very fast, switching every few time steps from one CA to another, but it can also need hundreds of time steps before any shift occurs. The number of cell assemblies seems to have an effect on the dynamics as well; all observed dynamics (in the cell assembly, the whole system or spontaneous activity) were reinforced by the number of cell assemblies present in

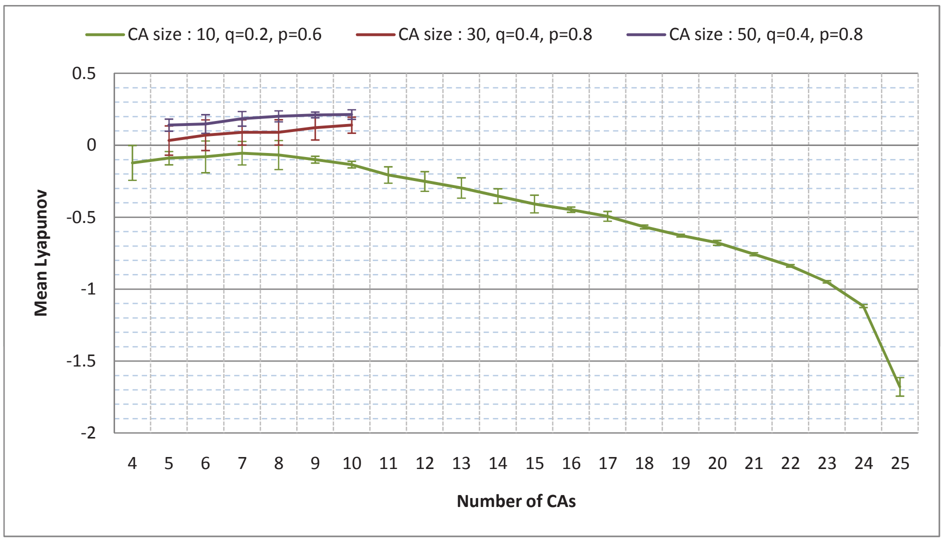


FIGURE 6.10. Dynamics of the system during stimulus for large data sets. Shown as the mean Lyapunov activity the system during stimulus. The test is done for 100 random networks. Different configurations of cell assemblies are tested; the number of cell assemblies present in the system is plotted on the x-axis. In the first configuration (in green) the cell assemblies have 10 units ($q = 0.2, p = 0.6$); in the second configuration (in red) they have 30 units ($q = 0.4, p = 0.8$); and finally in the third configuration (in purple) they have 50 units ($q = 0.4, p = 0.8$). The number of cell assemblies in the system enforces the type of dynamics observed when having only two cell assemblies. Smaller cell assemblies become more and more stable, while the opposite trend is observed for larger cell assemblies.

the system. Again, large overlapped cell assemblies become more and more chaotic where small weakly overlapped cell assemblies become extremely stable when their number increases in the system. As a direct effect of this, the probability of observing such complex dynamics is correlated in a similar manner to the size, number and overlapping of cell assemblies.

In conclusion, this chapter proposed a simple model to implement a working memory, which successfully exhibits the required features for such a memory, based on a priori definition of the system. This model is built on very well known component but the architecture is new, as is its use as a working memory [Salihoglu et al., 2009b].

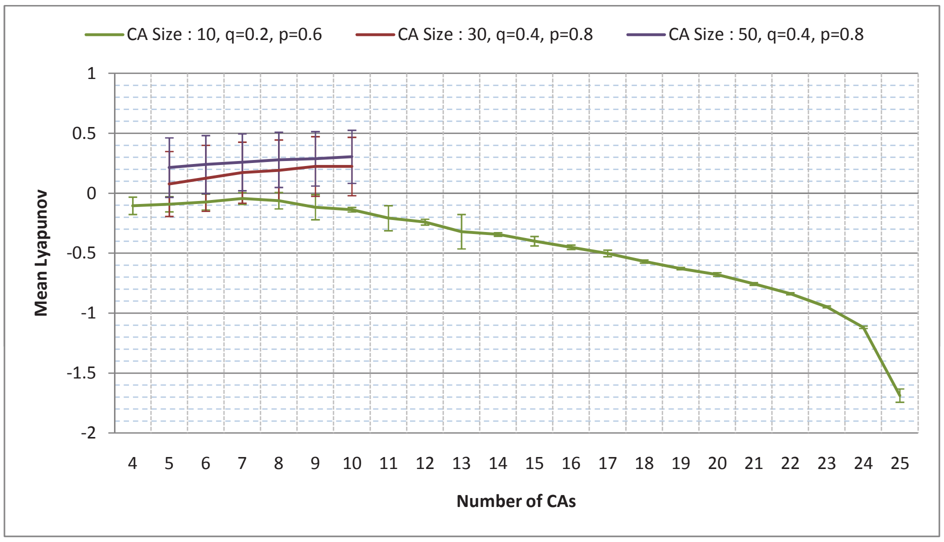


FIGURE 6.11. Spontaneous dynamics of the system for large data sets. Shown as the mean Lyapunov activity of the spontaneous activity of the system. The test is done for 100 random networks. Different configurations of cell assemblies are tested; the number of cell assemblies present in the system is plotted on the x-axis. In the first configuration (in green) the cell assemblies have 10 units ($q = 0.2, p = 0.6$); in the second configuration (in red) they have 30 units ($q = 0.4, p = 0.8$); and finally in the third configuration (in purple) they have 50 units ($q = 0.4, p = 0.8$). The number of cell assembly in the system enforces the type of dynamics observed when having only two cell assemblies. Smaller cell assemblies become more and more stable, while the opposite trend is observed for larger cell assemblies.

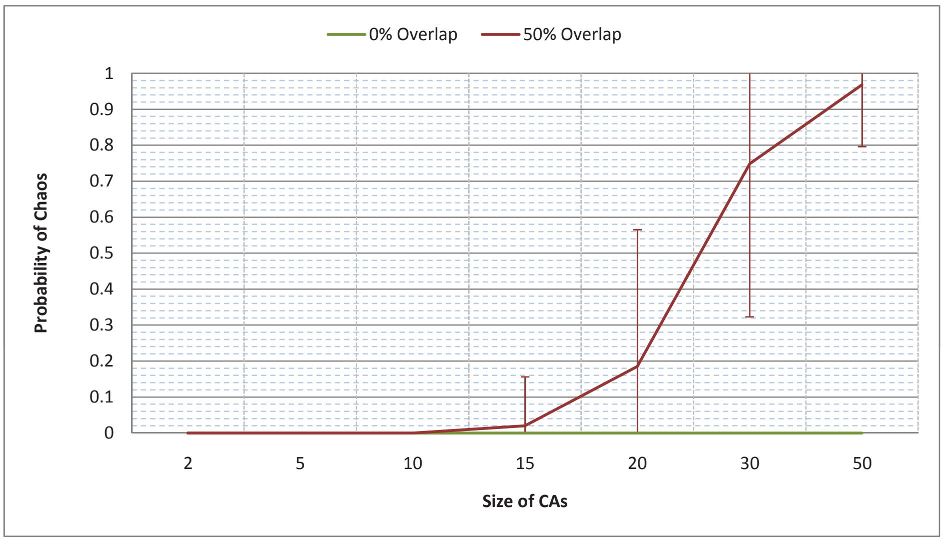


FIGURE 6.12. Probability of chaos in the system. The tests are done for 100 random networks, and 100 random inputs are tested for each. The system has two cell assemblies of various size and with two overlapping configurations. When there is no overlapping ($p = q = 0$) the system never exhibits chaotic regime (shown in green). However, when the overlapping is high ($p = q = 0.5$) the probability to find chaotic dynamics tends to one with the increasing size of the cell assemblies (shown in red).

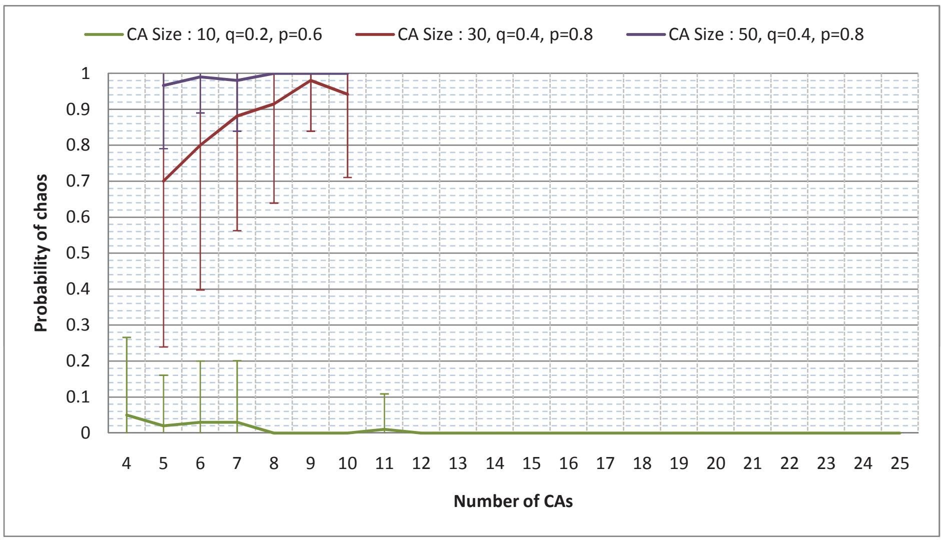


FIGURE 6.13. Probability of chaos in the system for large data sets. The tests are done for 100 random networks, and 100 random inputs are tested for each. Different configurations of cell assemblies are tested, the number of cell assemblies present in the system is plotted on the x-axis. In the first configuration (in green) the cell assemblies have 10 units ($q = 0.2, p = 0.6$); in the second configuration (in red) they have 30 units ($q = 0.4, p = 0.8$); and finally in the third configuration (in purple) they have 50 units ($q = 0.4, p = 0.8$). The small cell assemblies stabilize the networks, while large cell assemblies reinforce the presence of chaos, which becomes nearly omnipresent in the system.

CHAPTER 7

Online Adaptation Process

The previous chapter introduces the working memory models and the cell assemblies as a substrate for information encoding in those memory models. It has been shown how the model proposed in this thesis can effectively support the essential features of a working memory. The intuitive, a priori, definition of cell assemblies led to good results on this model. The goal of this section is to provide a learning mechanism that can bring such configurations in the system autonomously, in reaction to the stimulus needed to be learned.

Based on biological facts and the experience gathered from the development of various iterative Hebbian learnings, several features can be expected from this learning process. First, it must support asymmetric weights, since they promote complex dynamics and those dynamics are backed up by biological evidences as well as empirical tests done with this model in previous chapters. Secondly, the process needs to be autonomous, online and unsupervised. This second set of constraints came from several observations.

Indeed, the less specific the learning[1] is, the less the network gets constrained and thus better are the results; in addition, when talking about storing memories in the brain, it is hard to see an external supervisor validating the learning. This justifies the choice of unsupervised algorithms. The autonomous and online features come from the fact that the memory system can be enhanced by rehearsal or attention; but even without them, it is still very much active and most of the time processes information and learns the information without explicit consent (this is especially true for shorter memory).

Finally, an improvement of this model is suggested through a context layer which will helps the system resolve ambiguous information by providing contextual information for the stimuli.

It is important to note that unlike the previously described learning algorithms in Chapter 4 on page 81, here the learning is an online unsupervised process and thus does not have a data set to learn. The algorithms presented here are always active and react to the activity of the network without an a priori given definition of a learning task.

1. Network Plasticity: Creation of cell assemblies

The goal here is to provide an algorithm for the associative layer's synaptic weights ($\mathbf{W_R}$) plasticity (see Figure 6.1 on page 139). When an external stimulus is presented to the network, either the network's dynamics settles down to a specific attractor confined in a subset of cells, either the network will try to create a new cell assembly. Hebbian learning is a natural choice for cell assemblies formation and thus it has been chosen here as well: a simple, Hebbian-based, unsupervised rule is proposed for the creation of that cell assembly. This rule will apply to the

[1] In terms of the internal representation of the data that must be learned.

recurrent connections (the matrix $\mathbf{W_R}$ in Eq. 44 on page 78) and corresponds to a fast learning process (applied at each time step).

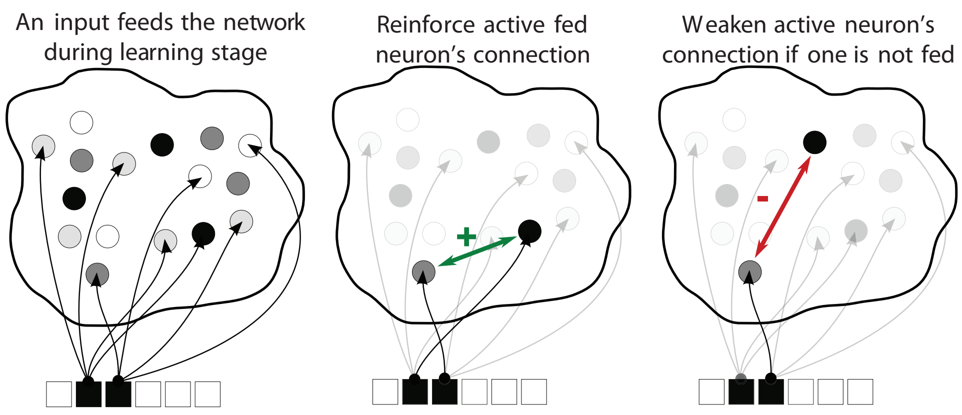

An input feeds the network during learning stage
Reinforce active fed neuron's connection
+
Weaken active neuron's connection if one is not fed
-

FIGURE 7.1. Basic steps of the Hebian/anti-Hebbian learning algorithm's implementation.

The basic idea behind this algorithm (illustrated in Figure 7.1) is that when an external stimulus is presented to the network, at each time step, some of the cells are picked up and their connecting synaptic weights are updated according to a Hebbian or an anti-Hebbian rule. If both units receive sufficient input stimulations, a Hebbian rule is applied to reinforce their connecting weights. On the contrary, if only one of the two units receives sufficient input stimulation, an anti-Hebbian rule is applied to promote inhibition between them. Algorithm 5 on the next page describes the learning in details.

Initially, the weight matrix $\mathbf{W_R}$ is initialized with a random uniform distribution in $[-0.05, 0.05]$, to create a random initial condition; yet the weights are small enough to avoid a major impact. In a random weight matrix, complex dynamics can be found but, since the weights are weak, the output will be weak too. The inhibition weights are kept as before ($\mathbf{W_A} = 1$, $\mathbf{W_I} = -1$)

Basically, this algorithm can be divided into three parts. First, cells that will be affected by the Hebbian learning mechanism are identified. This puts this algorithm aside from the two iterative Hebbian learnings proposed earlier (in Chapter 4 on page 81) since here the learning does not need to modify all the connections at each time step. The set of selected cells at time step t is noted $\mathbf{A}(t)$. Depending on the activity of the cells and on their connections with the input, this set can be further subdivided into four subsets depending on whether they are fed by the stimulus[2] or not, and they are active[3] or not. These four subsets are noted: $\mathbf{A_{fa}}(t)$, $\mathbf{A_{fi}}(t)$, $\mathbf{A_{ua}}(t)$, $\mathbf{A_{ui}}(t)$ (see the Algorithm 5 on the next page for more details on those subsets).

The second part of the algorithm consists in the critical part of weight modification. The weight modification is performed between all the units in $\mathbf{A}(t)$. The exact modification value (Δw_{ij}^R) for a connection between two cells depends on which subset each cell belongs to (see Algorithm 5 step 2)

[2]A cell is fed by the stimulus if the incoming signal to that unit from the input layer divided by the size of the stimulus is positive. The size of the stimulus is given by the number of units that are active in the input layer when that stimulus feeds the system.

[3]A neuron is said to be active if $x_i\,(t)$ is greater than some threshold (here 0.6).

Algorithm 5 Cell assembly creation. Hebbian plasticity of the recurrent connections (matrix $\mathbf{W_R}$ in Eq. 44 on page 78)).

Require:

At each time step, a set of k units are randomly picked up(here $k = 15\%N$). Among this set, named $\mathbf{A(t)}$, four types of cells can be identified depending on their activity and on their connection with the external stimulus:

- $\mathbf{A_{fa}}(t)$: active units which are fed by the input
- $\mathbf{A_{fi}}(t)$: inactive units which are fed by the input
- $\mathbf{A_{ua}}(t)$: active units which are not fed by the input
- $\mathbf{A_{ui}}(t)$: inactive units which are not fed by the input

```
1. for all $i, j \in \mathbf{A}$ do
2.     $\Delta w_{ij}^{R}$ is modified according to the following table:
```

i/j	f.a.	f.i.	u.a.	u.i.
f.a.	$\Delta(i,j)$	$\Delta(i,j)$	$-\Gamma(i,j)$	$\Omega(i,j)$
f.i.	$\Delta(i,j)$	$\gamma\Gamma(i,j)$	0	$\Omega(i,j)$
u.a.	0	$-\Gamma(i,j)$	$-\Gamma(i,j)$	$\Omega(i,j)$
u.i.	$\Omega(i,j)$	$\Omega(i,j)$	$\Omega(i,j)$	$\Omega(i,j)$

Where:

- $\Delta(i,j) = \delta(w_{ij})\,\mathrm{sgn}(w_{ij})$
- $\Omega(i,j) = \begin{cases} -\delta(w_{ij})\,\mathrm{sgn}(w_{ij}) & \text{if } |w_{ij}| > \omega \\ 0 & \text{otherwise} \end{cases}$
- $\Gamma(i,j) = \delta(w_{ij})$
- $\delta(w_{ij})$ is uniformly sampled in the range $[0, \varepsilon(1 - 0.9|w_{ij}|)]$
- Here the learning parameters are $\varepsilon = 0.05$, $\gamma = 0.02$ and $\omega = 0.2$

```
3. end for
4. for all neuron $i$ do
5.     compute its incoming and outgoing connections:
       $IN_i = \sum_j |w_{ij}^R|$ and $OUT_i = \sum_j |w_{ji}^R|$
6.     If $IN_i > \mu N$, rescale all $w_{ij}^R$ such as $IN_i = \mu N$
7.     If $OUT_i > \mu N$, rescale all $w_{ji}^R$ such as $OUT_i = \mu N$ (here $\mu = 0.95$)
8. end for
```

The first obvious observation in this table of modification is that the table is not symmetric, which suggests that the network will end up with asymmetric weights. Another indication comes from the random part of the $\delta(w_{ij})$ function which will give different results even for symmetrical weight modifications. In addition to the randomness, this function is also dependent on the weight of the connection, thus the less symmetric the weights are, the less likely $\delta(w_{ij})$ and $\delta(w_{ji})$ will be close to each other.

The table contains three main weight modification functions: $\Delta(i,j)$, $\Omega(i,j)$ and $\Gamma(i,j)$.

The first function $\Delta(i,j)$ enforces a connection, which means that $\left|w_{ij}^R + \Delta w_{ij}^R\right| \geq \left|w_{ij}^R\right|$. The idea behind this function is to improve whatever influence one cell had on the other, independently of the nature of the connection (inhibitory or excitatory). This function has two components: $\delta(w_{ij})$ and $\mathrm{sgn}(w_{ij})$. The first component is

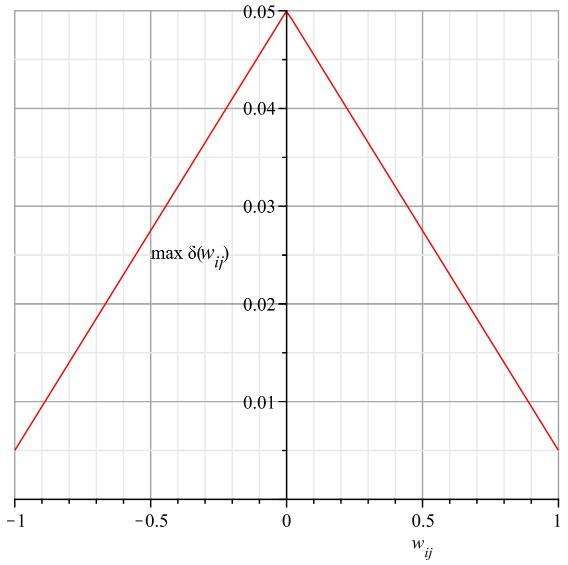


FIGURE 7.2. Upper bound of $\delta\left(w_{ij}\right)$. The y-axis shows the weight applied on $\left(w_{ij}\right)$.

a random function that fixes the actual amplitude of the modification. The randomness is used to produce asymmetric weights as explained before. The value is bounded in $[0, \epsilon\left(1-0.9\left|w_{ij}\right|\right)]$. The upper bound of this range reflects several important features. First, the stronger the weight, the smallest the maximum modification (this is given by $1-0.9\left|w_{ij}\right|$). Secondly, this bound scales the weight modification in a smaller range, which avoids brutal weight modifications. This is done by the learning parameter $\varepsilon = 0.05$. The function $\delta\left(w_{ij}\right)$ can be seen in the Figure 7.2. The second component is obvious and guarantees that the sign of Δw_{ij}^R will be the same as the sign of w_{ij}^R.

The second function $\Omega\left(i,j\right)$ is closely related to the $\Delta\left(i,j\right)$ function, but weakens the existing connection between two cells. It also uses the $\delta\left(w_{ij}\right)$ function and the sign of w_{ij} but this time there is a -1 factor. In other words, a positive connection becomes less positive and a negative connection becomes less negative. An important component of this $\Omega\left(i,j\right)$ function is that it has no effect if the weight of the connection is not strong enough ($\left|w_{ij}\right| > \omega = 0.2$). This function is used in order to weaken a connection between two unrelated cells without completely dampening the existing connection too much. The third function $\Gamma\left(i,j\right)$ is just an other notation for the $\delta\left(w_{ij}\right)$ function.

The function $\Delta\left(i,j\right)$ is used to reinforce the connections between active fed neurons and other fed neurons. The logic behind this is that all the neurons that are fed by the stimulus are probably concerned by the stimulus and thus must be active. Hence, the connection between all the fed neurons with an active fed neuron is reinforced in order to promote a stronger activity of all those neurons.

The function $\Omega\left(i,j\right)$ is used to weaken the connection between inactive unfed neurons and any other neurons in $\mathbf{A}(t)$. The reason why this weakening is bounded by $\omega = 0.2$ is to take into account the fact that two unfed inactive neurons may have a strong connection because they are fed by another stimulus under which they are active and thus strongly connected.

The use of the function $\Gamma(i,j)$ is a bit more subtle. It is used to decrease the connection between two unfed active neurons (by using $-\Gamma(i,j)$). The difference between this function and the function $\Omega(i,j)$ is that this function does preserve a minimal weight (ω) and the function $\Omega(i,j)$ is dependent on the sign of the weight. Thus when the weight is negative it will make it stronger, where $-\Gamma(i,j)$ will make it even more negative. Basically, $-\Gamma(i,j)$ tends to make two neurons inhibit each other while $\Omega(i,j)$ tends to make them ignore each other. Naturally, when two neurons are active but unfed, it seams natural to try to shut them down.

On the other hand, if two neurons are fed but inactive it is a good idea to try to reinforce their connection, thus, the function $\gamma\Gamma(i,j)$ is used. This function differs from $\Delta(i,j)$ since $\Delta(i,j)$ would make a negative connection even more negative, where this function always tries to make it positive. Additionally, the learning factor $\gamma = 0.02$ is applied in order to keep a difference in the order of magnitude between the kind of reinforcement that is produced when adjusting the weights between two active fed neurons and two inactive fed neurons. This is important because if the neurons are inactive, maybe they do not belong to the response to that stimulus and thus should not be made active by reinforcing too much the connection between them. The idea here is to give those neurons a little push: if it helps and one of them becomes active then, the next time they are considered, they will receive a strong reinforcement.

The function $-\Gamma(i,j)$ is also used for the connection going from an unfed active neuron to a fed active neuron and from a fed inactive neuron to an unfed active neuron. The first case comes from the fact that if a neuron is unfed it does not belong to the current cell assembly, and moreover it is probably part of another cell assembly. As with the case of two unfed active neurons, the safest approach for this unfed neuron is to inhibit the active and fed neuron (as cells from a different assembly do). Incidentally, this will increase the connection amongst neurons of the active cell assembly, since the impact of the unfed but active neuron is diminished and thus, in the next step, the fed neuron will need to be increased in order to be kept active. However, the active and fed neuron does not change anything to its connection towards an active but unfed neuron: since there is no guarantee that this neuron really belongs to a cell assembly, the learning cannot be sure that it needs to weaken this connection (as it did in the other direction). If this neuron was part of another cell assembly then, when this cell assembly is activated the connection can and will be weakened, because this time the neuron that was fed in the first case will now be an unfed neuron.

Finally, the function $-\Gamma(i,j)$ is used to weaken the connection from a fed inactive neuron to an unfed active neuron. Since the neuron is inactive this means it does not affect much the unfed active neuron. The connection is weakened in order to slightly reduce the impinging strength to the unfed active neuron. It is useful to shut down unfed neurons that were barely active and thus should probably not have been active to begin with. On the other hand, the connection from an unfed active neuron to a fed inactive neuron is left untouched. This can be understood as a way to help the fed inactive neuron to become active. Since the neuron is not active and needs to be (since it is fed), it means that it does not receive a strong enough signal from other neurons. So weakening a connection from an active neuron (even unfed) will certainly not help. Yet, this connection cannot be reinforced since the neuron is unfed and, as shown with a connection going from unfed active neuron to fed active neuron, a connection from an unfed active neuron to an neuron that belongs to the response of the current stimulus must be kept weak.

The final and last part of the algorithm consists in re-normalizing the weight matrix. Given the upper bound of the $\delta(i,j)$ functions, the weights of a single unit cannot go above $\dfrac{10}{9}$ and thus there is no need to take care of individual weights. Yet, it is important that all the connections from (or to) a single unit are not all strong. This re-normalization modifies the weights such that the total synaptic weights is bounded in order that the sum of the outgoing (or incoming) connections of any unit is below a given threshold (here $0.95N$). This promotes individuality of neurons since they cannot be strongly related to all the other ones.

2. Cell assembly formation

This Hebbian learning (described in Algorithm 5 on page 155) is tested here as a working memory. The feed forward connections from the stimulus to the associative layer ($\mathbf{W_S}$) are randomly initialized in $[0.01, 0.3]$ and do not evolve.

Figure 7.3 shows the weight matrix $\mathbf{W_R}$ after learning 10 different stimuli. Each stimulus is presented two times to the system for 200 time steps. For all the learned stimuli the same cell assembly is detected as active. In this figure units of the associative layer have been reordered so that the ones that belong to the CA appear first (on the figure they are highlighted with the blue boxes). This shows that the Hebbian learning alone failed to create the CAs. The reason is the following. No constraint is put on the units' choice during the learning procedure (Section 1 on page 153): the algorithm tries only to maintain the activity of a subset of cells. As a result, the easiest solution for the Hebbian procedure is to converge to a unique response to all stimuli. Said differently, the only way to stabilize the newly formed CA without destroying the previously created ones is to integrate it with them, ending in one big CA, which reacts similarly for all the stimuli. As a result, the weight matrix obtained after the encoding process and after reordering the recurrent units according to their activity, is composed by only one big CA.

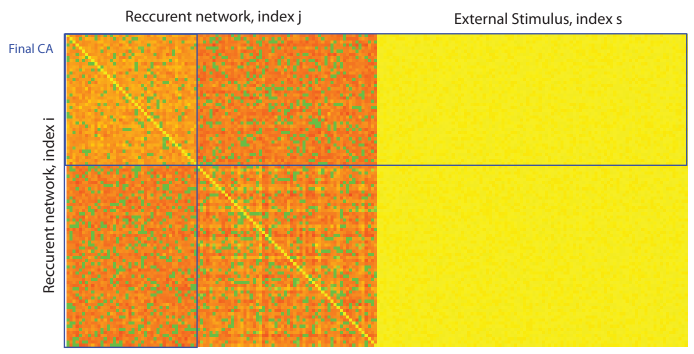


FIGURE 7.3. Impact on the weights of the Hebbian learning. A posteriori observation of the cell assemblies in the synaptic weights of the recurrent and feed forward connections when no retroaxonal procedure is involved. Each row corresponds to the connections impinging one unit (from left to right, the recurrent then the feed forward connections). The recurrent matrix was reordered according to the identified cell assemblies, here one big CA covering half the network.

This result helps to set up the next step for this algorithm to be functional. The way the weights of the associative layer evolve is deeply related to neurons being fed or not by a stimulus during the learning process. Yet here the connection between the input layer and the associative layer is static. The other important consideration concerning those connections is that they are the ones dictating the learning algorithm which neurons should be clustered together, and thus the random initialization of those weights is unlikely to lead to any significant result. This is why, in the next section, a more subtle process is introduced in order to help the Hebbian process.

The next section will introduce a recent popular hypothesis called the "retroaxonal hypothesis" which is used in this thesis as a complementary process to the Hebbian learning in order to help it create the cell assemblies.

ε	0.05	It is chosen quite small in order to avoid brutal changes. From experimentation, a smaller value slows the learning process too much, and a larger value induces too strong changes.
γ	0.02	Chosen very small in order to keep the total secondary learning $(\gamma \cdot \varepsilon)$ much smaller than ε
ω	0.2	Avoid weakening process to dampen too much the existing weights
μ	0.95	Allow 95% of the units fed (or feeding) a given cell to be strongly connected $(w_{ij} = 1)$
$\lvert A\rvert$	100	Size of the associative layer.
$\lvert S\rvert$	100	Size of the input layer.

TABLE 1. This table recaps all the parameters used for the experiments in Section 2 on the facing page with their value.

3. The Retroaxonal Hypothesis

The most popular learning algorithm in neural networks is probably the back-propagation algorithm [Werbos, 1974]. This algorithm has seen a lot of successful real-world applications but never managed to convince the neuroscience community because of its requirement for information to go backward along axons (called retroaxonal signals). Yet, recent experiments from Du and Poo [2004] have shown that retroaxonal signals can indeed exert control over a cell's input synapses [see also Fitzsimonds et al., 1997; Tao et al., 2000]. This fact is very important for theories of neural information processing, because it suggests that retroaxonal signals are meaningful in the brain and might be able to organize the construction of the internal representations required for complex behaviors. Unfortunately, this does not validate the back-propagation algorithm: while in the algorithms where the retroraxonal signals are instantaneous, in the brain they are slow (an order of magnitude slower than synaptic timescales).

The first evidence that neurons are capable of carrying retroaxonal signals came from experiments showing that the survival of developing spinal neurons depends on signals received from their target muscles [Hamburger, 1992, 1993]. Later, it has been shown that the survival of many other cell types is influenced by retroaxonal signals [Buss et al., 2006; Oppenheim, 1991]. Retroaxonal signals control many developmental processes other than cell survival [Hippenmeyer et al., 2004; Kalinovsky and Scheiffele, 2004]. Moreover, several recent studies show that retroaxonal signals can indeed control the plasticity of input synapses [Du and Poo, 2004; Lom et al., 2002].

Harris [2008] proposed a framework in which retroaxonal communications occurring on realistic timescales could guide the formation of behaviorally adapted representations. His framework is based on the hypothesis that the strengthening of the output synapses of a neuron stabilizes recent changes in the same neuron's inputs. Here the retrograde signals are not instructive but selective: during learning, changes in neuron's input synapses lead to changes in its spiking pattern, which, in turn, may or may not lead to changes in output synapses. If the new spiking pattern promotes the formation or strengthening of outputs, the recent changes to the neuron's input synapses are retained; if not, the changes decay (see Figure 7.4).

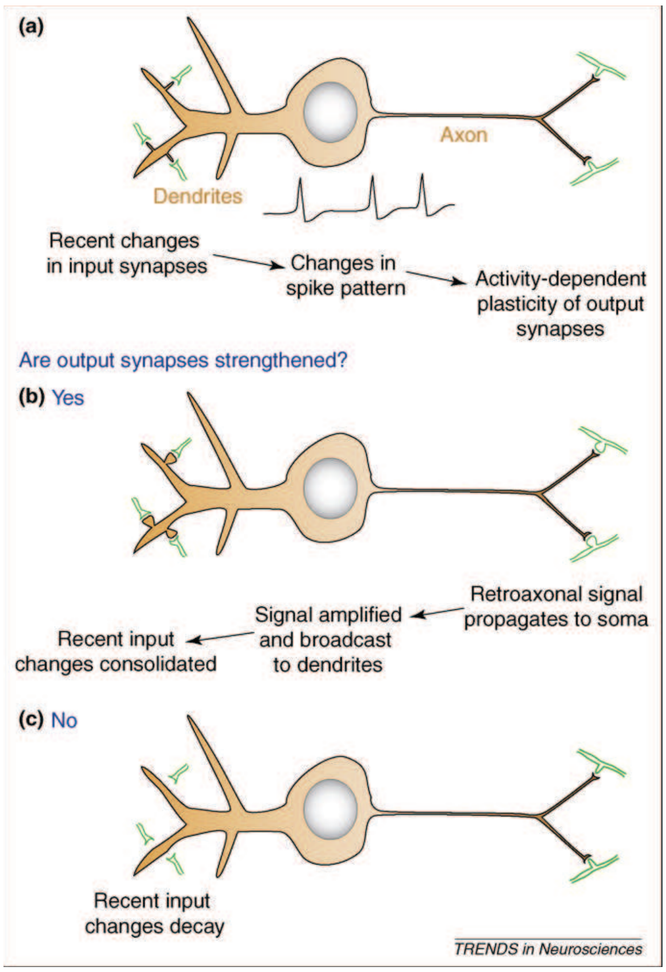

(a)
Axon
Dendrites
Recent changes in input synapses
Changes in spike pattern
Activity-dependent plasticity of output synapses
Are output synapses strengthened?
(b) Yes
Retroaxonal signal propagates to soma
Signal amplified and broadcast to dendrites
Recent input changes consolidated
(c) No
Recent input changes decay
TRENDS in Neurosciences

FIGURE 7.4. The retroaxonal hypothesis: strengthening of a neuron's output synapses stabilizes recent changes in the same neuron's inputs. (a) Changes in a neuron's input synapses lead to an alteration in its spiking pattern. This, in turn, may or may not lead to plasticity of output synapses. (b) If the neuron's output synapses are strengthened, this initiates a retroaxonal signal that causes consolidation of the recent changes in inputs; changes in the neuron's spike pattern therefore become permanent. (c) If output synapses are not strengthened, the retroaxonal signal is not received, the recently changed input synapses decay to their prior state and the neuron reverts to its original spiking pattern.

To further understand the intuition behind this hypothesis, it is important to look at the stability of hippocampal place fields. Frank et al. [2004] have conducted an experiment in which they recorded CA1 pyramidal neurons while rats explored an environment which was largely familiar, except for one novel region that was previously inaccessible. The goal of this experiment was to study the formation of place fields. Interesting results have been reported when the rats were allowed to explore the novel region, place fields formed within minutes, and on the first day of recording there were approximately twice as many place fields coding for the novel region as the familiar part of the apparatus. However, after three days of exploration, the density of place fields became uniform amongst all regions.

This raises the question on what mechanisms are responsible for the stabilization of a place field? Some studies with mice [Agnihotri et al., 2004; Hollup et al., 2001; Kentros et al., 1998, 2004] suggest that the stability of a place field might correlate with the utility of the information it carries for performance of behaviors. Based on these evidences, Harris [2008] posits that retroaxonal signals may be the mechanisms behind this feature.

This thesis took a particular interest in this hypothesis for two reasons. First, as suggested by Harris [2008], this hypothesis can be suitable for stabilizing the formation of cell assemblies. Secondly, if applied successfully, this work can give a little more incentive and credibility to the hypothesis which deserves more interest from researchers.

3.1. 'Retroaxonal' signals stabilizing the cell assemblies. To help the Hebbian Algorithm 5 on page 155, the retroaxonal hypothesis is used to continuously stabilize the cell assemblies by the input system. To this aim, the feed forward input connections (matrix $\mathbf{W_S}$ in Eq. 44 on page 78) have to be learned to acknowledge the creation of the new cell assemblies. To get rid of the fluctuations of the network's complex dynamics, this learning occurs at a slower timescale. Here, the retroaxonal procedure is arbitrarily chosen to work twenty times slower than the Hebbian process.

Algorithm 6 Cell assembly stabilization. Retroaxonal signals for learning the feed forward connections (matrix $\mathbf{W_S}$ in Eq. 44 on page 78)).

```
1. Every $T$ time step (here $T = 20$)
2. for all unit $i$ do
3.         Compute the average activity of unit $i$ over the last $T$ time steps (from
           time $t_s$ to time $t_e$):
(74)           $\bar{x}_i = \frac{\sum_{k=t_s}^{t_e} x_i(k)}{T}$
4. end for
5. for the $k$ most active units in average, (here $k = 5$) do
6.         if $\bar{\mathbf{x}}_\mathbf{i} >$ some threshold (here 0.2) then
7.                 $\Delta w_{is}^S = \begin{cases} \nu & \text{if } \iota_s \text{ is active} \\ -\nu & \text{otherwise} \end{cases}$
                   Where $\nu$ is the retroaxonal learning rate (here 0.05). $w_{is}^S$ is always
                   kept in $[-0.075, 0.3]$.
8.         end if
9. end for
```

Algorithm 6 describes the retroaxonal learning in details. Since the goal of this learning is to enhance selectivity in the feed forward connections, the intuition is that the newly formed cell assembly has to instruct the feeding network (here $\mathbf{W_S}$) through a feedback mechanism. Initially, the weight matrix $\mathbf{W_S}$ is randomly initialized in $[0.01, 0.1]$. The distribution is chosen to be weak (< 0.1) in order to let the learning process do most of the work and not to influence too much the system with the initial distribution. The connections are of course positive; otherwise they would be disabled and this is not desirable since, if too many connections get disabled, the system will get disconnected from the input layer.

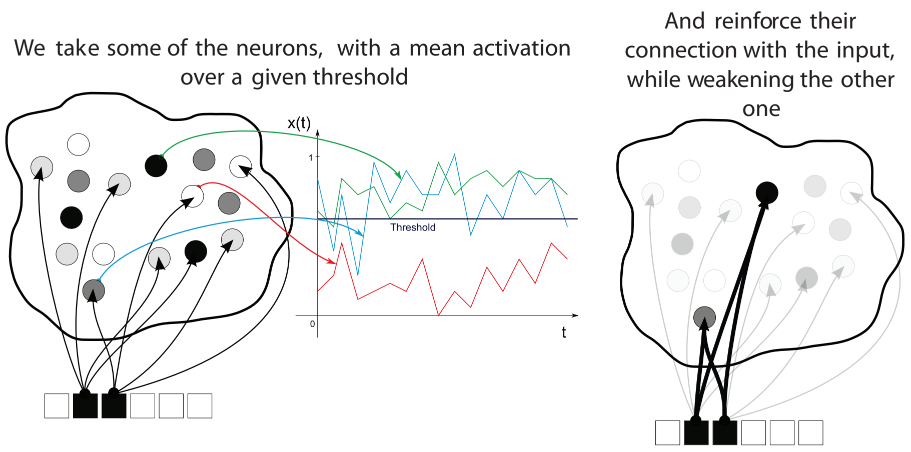

We take some of the neurons, with a mean activation over a given threshold
x(t)
1
Threshold
0
t
And reinforce their connection with the input, while weakening the other one

FIGURE 7.5. Basic steps of the retroaxonal learning algorithm's implementation.

This algorithm can be divided into two parts: first, it computes the average activity of each unit since the last time it was applied (here every $T = 20$ time steps); secondly, it uses this information to mimic the retroaxonal signals for the k most active units (Figure 7.5 shows the basic idea behind this implementation.). The first part is pretty obvious and simply consists in computing the average activity of a unit over the last T time steps (see equation 74 on the previous page). The second part first involves selecting the k most active units. Here k has been set to 5, to avoid too many units from the input layer to be associated to any unit from the associative layer in a single retroaxonal pass. After the units have been selected, if they show some noticeable activity (above 0.2), they receive a retroaxonal feedback. In the model proposed for this thesis, the retroaxonal feedback from a unit i (in the associative layer) to unit s in the input layer is the modification of the weight w^S_{is}. The way this weight is modified depends on the nature of the input unit s. If the unit was active ($x^S_i(t) > 0$), then it is considered partly responsible for the activation of the associative layer unit i and thus is a valid candidate for the retroaxonal feedback. Therefore, the weight w^S_{is} is reinforced by the learning factor $\nu = 0.05$. If the unit s was not active, it receives a negative feedback. This negative feedback somehow plays the role of the decay on the weight and is mainly used to create a competition amongst input layer's units. With this competition, it is unlikely to have to many different inputs being strongly connected to the same unit.

3.2. The Two Learning Algorithms Put Together. These two learning processes (Algorithms 5 on page 155 and 6 on the previous page) will occur only when external stimulation is applied, which could reflect some attention mechanism. No learning occurs in the absence of external stimuli, but since this would prevent offline consolidation, this feature could be added in future works.

For an external stimulus $S_{1,\ldots,M}$ and a unit i in the associative layer, depending on the unit's past averaged activity and on the level of inputs it received from individual synapses $(w^s_{ik}I_k)$, three different scenarios can occur.

(1) **The unit was active and received consequent inputs**. The unit is said to be part of the current cell assembly and that trend will be reinforced by the two learning mechanisms.
(2) **The unit was not active and did not receive consequent inputs**. The unit is not part of the current cell assembly and will be shut down by both learning procedures.
(3) **The unit was active but did not receive consequent inputs**. In that particular case, two competitive processes occur. First, due to the anti-Hebbian learning rule applied (Algorithm 5 on page 155, Line 2 on page 155), its recurrent connections decrease, tending to exclude it from the current CA. Secondly, the retroaxonal signals reinforce the synaptic weights from its input connections, tending to include it in the current CA.

The main problem observed with the Hebbian algorithm when it was used alone was its inability to cluster the input space in order to avoid the formed cell assembly for each different input to overlap too strongly. As shown earlier, when the cell assemblies are too overlapped the Hebbian algorithm cannot manage to form new distinct cell assemblies for different inputs and the system ends up having a single cell assembly as a response to all inputs.

On the other hand, the retroaxonal feedback role is basically to cluster the feed forward connections coming from the input layer. This is exactly the missing component of the Hebbian learning. Additionally, as shown in the earlier sections, the Hebbian learning's ability to determine if two neurons should belong in the same cell assembly depends strongly on whether they are fed or not by the input layer. It is obvious that the retroaxonal feedbacks will change this information and thus the clustering appearing from this feedback signals will end up reflecting on the cell assemblies the Hebbian learning will be able to form.

To conclude, those two learning processes will be active during all the activities of the system and they will provide the evolution rules for the weights of the network in a similar way that the simulation rules provide the evolution for the network's states.

4. Results for Online Formation of CAs

4.1. The Simple Case.

4.1.1. *Encoding the cognitive maps.* Two cognitive maps (as defined in Section 5.3 on page 76), each one composed of 5 external stimuli, were presented and encoded in the network: each stimulus corresponds to a subset of 20 cells of the input layer set to 1, while the other 80 cells were set to 0.

External stimuli were set as follows. If the input vector is divided into 5 parts of 20 cells, stimuli of the first map consisted in the activation of four cells from each subset. For example, for the first stimulus, the first four cells from each subset were set to 1. For the second map, each subset was successively set to 1. Since the input patterns are strongly reflected in the feed forward connections by the learning process, the structure of the two maps appears in Fig. 7.7 on page 166.

During the encoding/learning phase, each stimulus from the first cognitive map was sequentially activated during 50 time steps. This process was repeated 10 times. A similar encoding was then applied to the second cognitive map. The whole procedure was carried out twice in order for the system to see both inputs more than once. This is not very relevant when dealing with only two inputs but when the number of inputs increases it becomes useful for the system to see an information more than once in order for it to encode it and manage to not forget it afterwards under the heavy learning pressure of a large array of stimuli. This is reminiscent of the rehearsal mechanism in working memories in order to maintain informations in the short-term memory store.

4.1.2. *Cell assembly identification.* When cell assemblies were a priori encoded in the network's recurrent connections as in Chapter 6, the cell assembly activity could easily be monitored by averaging the activity of all its constitutive units. In this section, the network is expected to dynamically create cell assemblies, and accordingly there is no a priori knowledge of its constitutive units. To identify a posteriori the cell assembly's constitutive units, two methods could be applied. First, cell assemblies could be detected from the passive observation of clusters in the recurrent weight matrix $\mathbf{W_R}$: strongly connected units will have a high chance to belong in the same cell assembly. Secondly, cell assemblies could be detected from their response to the previously applied external stimuli.

The latter procedure was applied here: for each external stimulus, the network's activity is monitored during $p = 100$ time steps. The cell assembly associated with the stimulus is defined by the set of all units having an average activity greater than a threshold value (here, 0.4). The reason behind this choice is that it is more accurate for the practical case, even though the first one is more likely to find good solutions in theoretical analyses: as shown earlier in this work, all strongly connected units do not always belong to the same cell assembly while sometimes weakly connected units can be part of a same cell assembly; this makes it very tricky to find a good algorithm which will lead to good results in practice. The problem with the second approach is that, since it only looks at the activity of the cells, it can miss some of the neurons that should belong to the cell assembly but are inactive for that particular input. However, if some units do not reactivate when the input hits the system, there is no practical use to consider it, even if it should theoretically belong to the cell assembly for some reason.

4.1.3. *Recall.* After the encoding phase, the created cell assemblies were detected by the method defined Section 4.1.2. Figure 7.6 on the next page shows the rasterplot of the network activity and of the average activity during and after periods of reactivation where each time 20% of an original stimulus fed the network.

To obtain a clear rasterplot activity in Fig. 7.6 on the facing page, units of the recurrent network were first reordered according to the detected cell assemblies. The activity of the newly created cell assemblies (Fig. 7.6 on the next page) appears very similar to the activity of the pre-encoded cell assemblies (Fig. 6.2 on page 140). In both cases, cell assemblies were successfully reactivated with noisy inputs stimulation (prerequisite for a content-addressable memory) and maintained their activity after stimulus offset (prerequisite for a working memory). This indicates that this algorithm seems a good candidate for encoding cell assemblies in the recurrent connections of the network.

Figure 7.7 on page 166 shows the synaptic weight matrices $\mathbf{W_R}$ and $\mathbf{W_S}$ (Eq. 44 on page 78). The cell assemblies identified from the network activity are highlighted in the figure; e.g. 'CA 1-CM 1' are the cells which were only activated when the stimulus one from the first cognitive map was applied. Shared units are the units which can be activated by different stimuli.

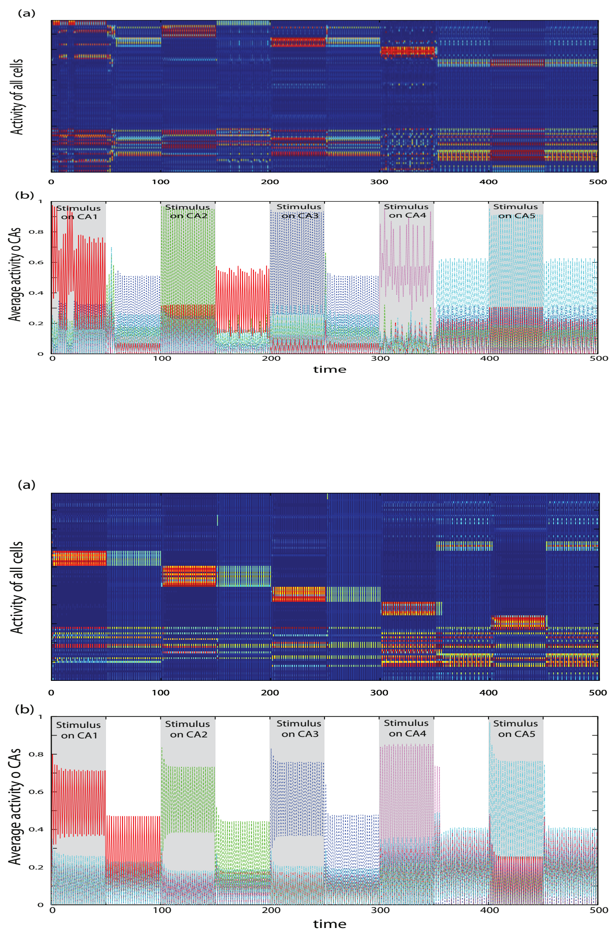


FIGURE 7.6. Content-addressable and working memory features after learning. The dynamically created cell assemblies for the first cognitive maps (top) and the second cognitive maps (bottom). Sequences of spontaneous activity following sequences of external stimulation were tested. (a) Raster plot showing the activity of the 100 cells of the network. The units of cell assemblies have been rearranged to appear contiguously on this raster plot and this makes it easy to detect the corresponding clusters of activity. (b) After identification of the cells constituting each CA, the activity through time of the 5 newly created cell assemblies is plotted.

The feedforward synaptic weights appear highly clustered by the learning procedure and reflect the external pattern applied to the network. As an example, the cluster 'CA 1-CM 1' receives strong inputs only from a well-defined subset of input units which are the units corresponding to the first pattern. On the contrary, shared or unused cells receive less clustered inputs. Clusters in the recurrent connections, while present, are still much more difficult to identify and to analyze.

4.2. Capacity. After these promising results, to have a better understanding of the features and limitations of these algorithms, capacity measures are performed here. Two important questions must be answered in order to understand the capacitive limits of this model: first, finding the maximum amount of information the

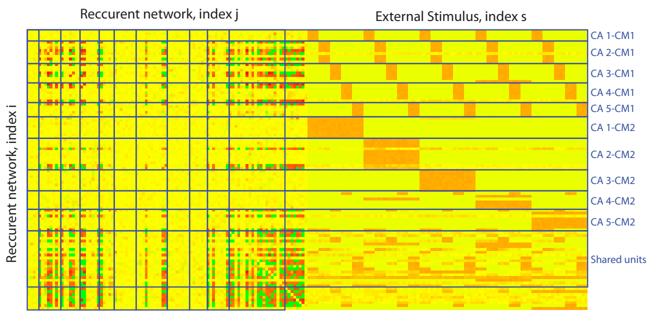


FIGURE 7.7. Weight matrix after learning. A posteriori observation of the cell assemblies in the synaptic weights of the recurrent and feed forward connections. Each row corresponds to the connections impinging one unit (from left to right, the recurrent then the feed forward connections). The recurrent matrix was reordered according to the identified cell assemblies. Cells belonging to more than one CA are put after all the other ones.

ε	0.05	It is chosen quite small in order to avoid brutal changes. From experimentation, a smaller value slows the learning process too much, and a larger value induces too strong changes.
γ	0.02	Chosen very small in order to keep the total secondary learning $(\gamma \cdot \varepsilon)$ much smaller than ε
ω	0.2	Avoid weakening process to dampen too much the existing weights
μ	0.95	Allow 95% of the units fed (or feeding) a given cell to be strongly connected $(w_{ij} = 1)$
ν	0.05	It is chosen quite small in order to avoid brutal changes. From experimentation, a smaller value slows the learning process too much, and a larger value induces too strong changes.
$\lvert A \rvert$	100	Size of the associative layer.
$\lvert S \rvert$	100	Size of the input layer.

TABLE 2. This table recaps all the parameters used for the experiments in Section 4.1.3 on page 164 with their value.

system can learn; second, understanding the consequences of crossing this critical limit. To do this, an increasing number of inputs are presented to the system. After the system has seen all of them, they are recalled. When the number of successful recalls cannot be raised by learning more inputs, the system is said to have hit its capacitive limit.

For this experiment an increasing number of inputs is proposed to the system and its performance was monitored. Both the associative and the input layers have 100 units. The inputs presented to the system are grouped into cognitive maps. A cognitive map represents a group of related inputs. Here, each cognitive map has five inputs of equal size and can be seen as a partition of the input layer into five equal sets of 20 units. Each input corresponds to one set of this partition, and the units of this set represent which unit of the input layer are active when this particular input feeds the system. For each input, the system is expected to map it to an original cell assembly. The inputs are generated randomly and presented in a random order. Once each input has been presented, they are proposed again but in a different order (each input is presented a total of three times). Figure 7.8 on the facing page shows the number of correctly learned inputs for a given number of cognitive maps. It appears that the system is not able to learn all the inputs when the number of cognitive maps increases, yet a large amount of information is learned. As with any memory-like structure, the system has a limit but, as shown by the figure, this is not a hard limit which causes catastrophic forgetting; instead, with the increasing size of information the system slowly forgets some of the inputs. Of course, rehearsal of an information will most likely make it active again but probably at the cost of some other information.

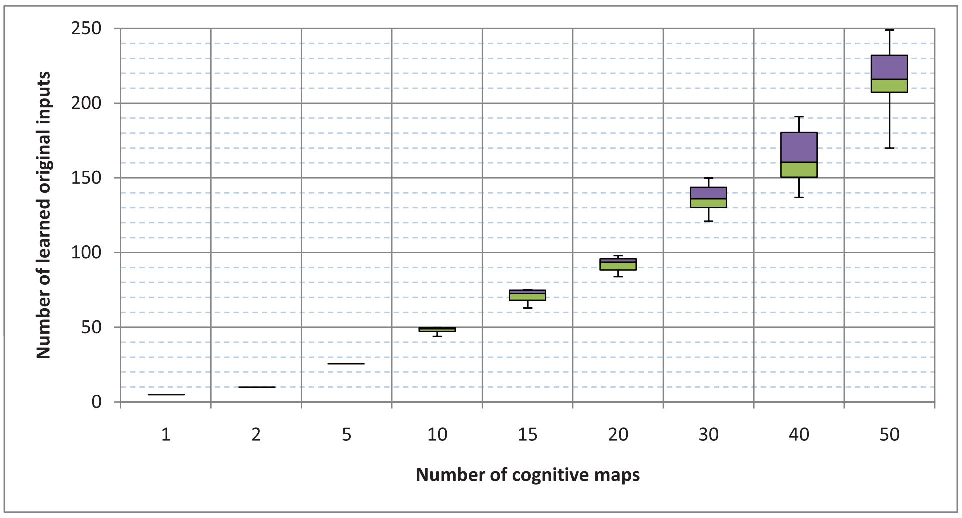


FIGURE 7.8. Capacity of the working memory. It clearly appears, on this box plot, that the system is able to learn an increasing number of cognitive maps composed of five inputs. The y-axis show the number of successfully learned inputs (the maximum being the cognitive map size times 5, since each cognitive map is composed of 5 inputs). However, when the data set increases, signs of capacitive limit are observed. This is a classic box plot representation: the line in the box is the median, the boxes represent the first (green) and third (blue) quartile and the dashes are the minimum and the maximum of the distribution.

Figure 7.9 on the next page shows the ratio between the number of inputs that are presented to the system and the number it is able to successfully recall. This plot makes it easier to see the forgetting that occurs in the system, as anything below 1 indicates that at least some inputs where not learned successfully or more exactly forgotten.

Even though the system learns a large number of information, it is important to note that at some point the system will interpret a supposedly different input as an already seen input. Given the number of partitions of 100 units into 5 sets of 20 units is finite and that each units is highly dependent on others units of the same partition, when the number of such partitions rises, the probability to find two similar inputs rises rapidly, too. In fact, when taking the noise tolerance of the system into account, two inputs do not need to be an exact match for the system to consider them equal. Figure 7.10 on page 169 shows the number of pairs of highly[4] correlated inputs found in random data sets depending on the size of the

[4]Here, two inputs having more than half of their units in common are considered highly correlated.

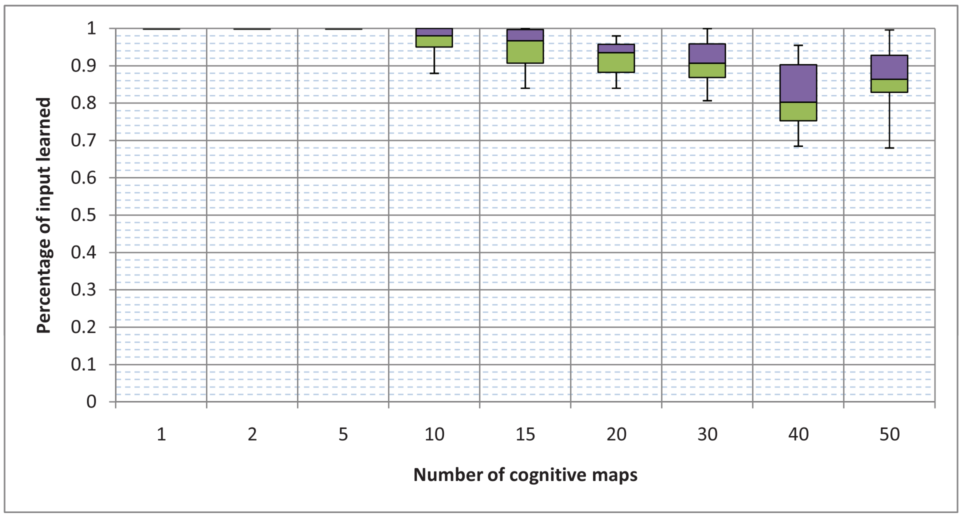


FIGURE 7.9. Capacity of the working memory (ratio). This box plot shows the ratio between the number of inputs presented to the system and the number of inputs actually learned by it. Even though the ratio is high, some information gets lost in the process and are forgotten by the system.

data set. As shown in this figure, this number rises rapidly with the data set's size. For instance there are 19 such pairs for 50 cognitive maps. To obtain 19 correlated pairs, the data set needs to have at least 6 highly correlated inputs. If there was only 5 highly correlated input, even if each one was highly correlated with all the other ones, the total number of correlated pairs will be 15. If the system has 38 highly correlated inputs, where for each input there is only one other input that is highly correlated with it, then there will be 19 highly correlated pairs in the system. This illustrates the bounds for the number of highly correlated inputs in the system for 50 cognitive maps and, as observed, this number is not negligible.

This observation also answers the second question about the capacity of the system: the system assimilates similar inputs as one, and thus uses already allocated memory places in its store for similar memories and avoids getting saturated with redundant data. Since the system does not seem[5] to show any testable limit it is hard to know what happens when this limit is crossed.

4.3. Noise tolerance. The capacity of a memory-like system is important, yet without noise tolerance it is not very useful. Most brain-like activities imply noisy information and thus require the system to be able to reconstruct the missing parts. For auto-associative memory, noise tolerance is even more important since it directly relates to the ability of the memory to have content-addressability. This section takes a closer look at the noise tolerance of the working memory model introduced in the previous chapter.

The tests are performed for different data set sizes (1,2,5,10,15,20,30,40,50) on a 100-neurons associative layer with a 100-neurons input layer. First the system is presented with the data set (three times), then the cognitive maps are learned (as shown in the previous section). After learning, inputs are present with varying

[5]The learning process, being online, takes a lot of time for large data sets and thus makes it hard to test this limit.

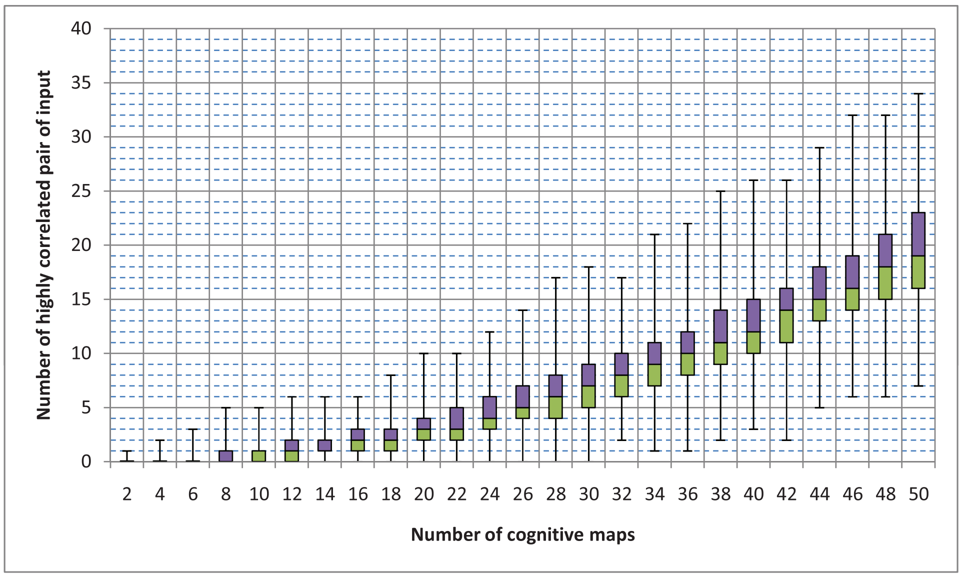


FIGURE 7.10. Highly correlated pairs of inputs present in a random data set. The number is computed for a data set of increasing size. The data set is composed of cognitive maps composed of 5 inputs. With the increasing data set size, more and more highly correlated inputs appear. Since the system is tolerant on noise, highly correlated patterns end up mapped to the same cell assembly.

ε	0.05	It is chosen quite small in order to avoid brutal changes. From experimentation, a smaller value slows the learning process too much, and a larger value induces too strong changes.
γ	0.02	Chosen very small in order to keep the total secondary learning $(\gamma \cdot \varepsilon)$ much smaller than ε
ω	0.2	Avoid weakening process to dampen too much the existing weights
μ	0.95	Allow 95% of the units fed (or feeding) a given cell to be strongly connected $(w_{ij} = 1)$
ν	0.05	It is chosen quite small in order to avoid brutal changes. From experimentation, a smaller value slows the learning process too much, and a larger value induces too strong changes.
$\|A\|$	100	Size of the associative layer.
$\|S\|$	100	Size of the input layer.

TABLE 3. This table recaps all the parameters used for the experiments in Section 4.2 on page 165 with their value.

noise applied to them and the system is monitored for its output. The system is said to have a successful recall when the expected cell assembly is the most active amongst all the cell assemblies present in the system[6].

The results can be seen in Figure 7.11 on the following page. The x-axis shows the amount of noise applied to the input in term of Hamming distance between the

[6]This implies that it must be active and more active than any other one.

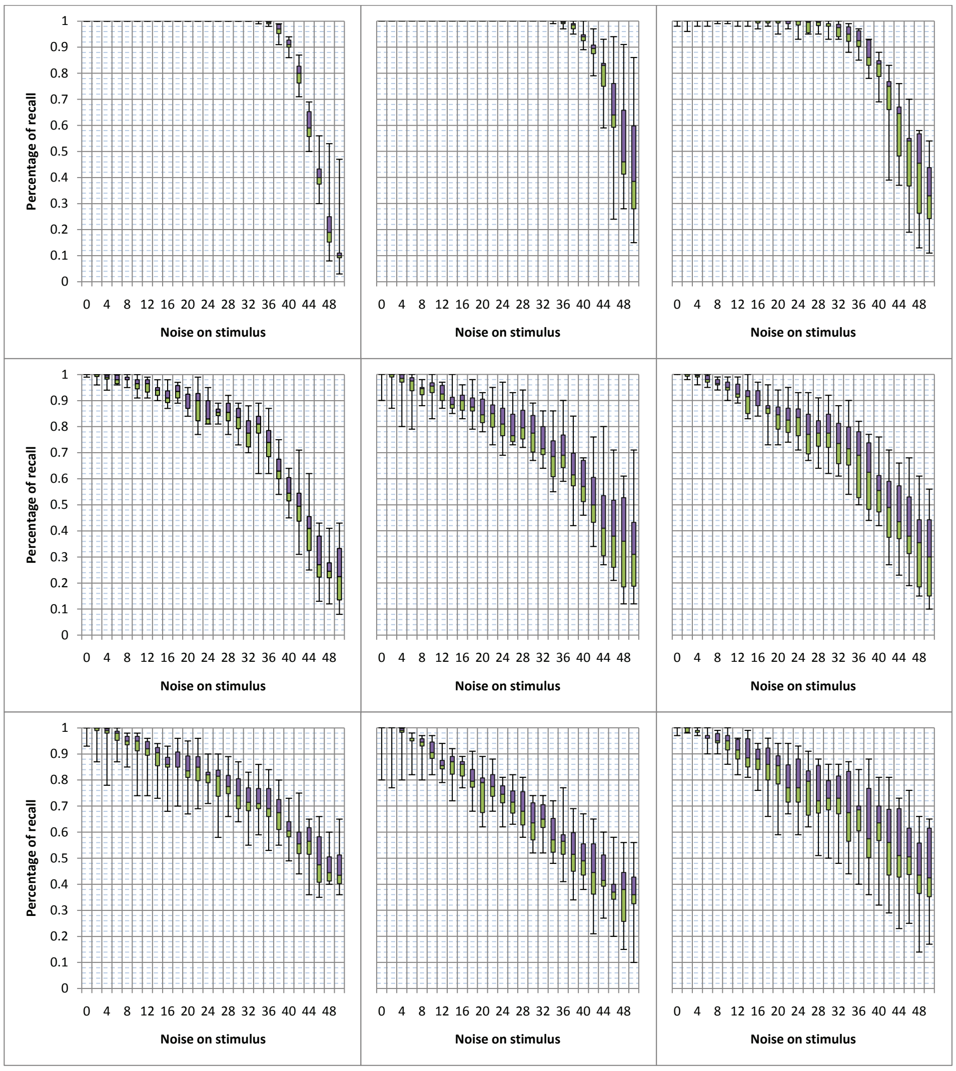


FIGURE 7.11. Noise tolerance. Shows the impact of noise on cell assemblies activation. It shows the percentage of successful recalls relative to increasing amounts of noise applied on the stimulus. The data set is composed of cognitive maps of 5 inputs. The input and associative layers have 100 units. Top Left: 1 CM, Top Middle: 2 CMs, Top Right: 5 CMs. Middle Left: 10 CMs, Center: 15 CMs, Middle Right: 20 CMs. Bottom Left: 30 CMs, Bottom Middle: 40 CMs, Bottom Right: 50 CMs.

correct input and the noisy version. The y-axis shows the percentage of successful recalls over a hundred trials.

From these results, it seems obvious that, even without noise when the data set increases in size, the correct cell assembly does not always have a successful recall. However, it is important to keep in mind that the learning procedures are unsupervised and were only asked to learn to have an active cell assembly in response to a set of inputs. Yet, for all these tests the cell assemblies have been detected using a method (see Section 4.1.2 on page 164) which does not guarantee a perfect detection. Therefore it is hard to know for sure if the system had really failed or if the detection process has failed and thus the wrong group of cells is observed. For example, a cell assembly may have some inactive neurons when fed with non-noisy stimulus, but those neurons gain strength when noise appears. In this instance, it is hard to decide how to exactly detect the cells belonging to the cell assembly. Another explanation of this behavior can be seen in the way the system handles ambiguity. Since the inputs are often correlated and even sometimes highly correlated (see Figure 7.10 on page 169), they naturally lead to overlapped cell assemblies. Therefore, even when only cells from one cell assembly activate, all the overlapped cell assemblies also show some activity. This can get even worse when the input suffers from noise which can lead to another overlapped cell assembly to become more active. This means that noise can easily transform an input into another one and thus what really happens is that the system gives the correct answer but not the one that would be expected from the un-noisy input.

ε	0.05	It is chosen quite small in order to avoid brutal changes. From experimentation, a smaller value slows the learning process too much, and a larger value induces too strong changes.
γ	0.02	Chosen very small in order to keep the total secondary learning ($\gamma \cdot \varepsilon$) much smaller than ε
ω	0.2	Avoid weakening process to dampen too much the existing weights
μ	0.95	Allow 95% of the units fed (or feeding) a given cell to be strongly connected ($w_{ij} = 1$)
ν	0.05	It is chosen quite small in order to avoid brutal changes. From experimentation, a smaller value slows the learning process too much, and a larger value induces too strong changes.
$\lvert A\rvert$	100	Size of the associative layer.
$\lvert S\rvert$	100	Size of the input layer.

TABLE 4. This table recaps all the parameters used for the experiments in Section 4.3 on page 168 with their value.

4.4. Dynamics. As in the previous chapter, the final step in the analysis of this memory is to take a closer look at the dynamics which appear in the system after the learning. During this experiment, the system learns a number of cognitive maps and then its dynamics are analyzed when different inputs feed the system. Figure 7.12 on the following page shows that these dynamics range from relatively stable dynamics to fairly complex ones and in average the Lyapunov exponents are near 0.0. First, it is important to note that the size of the data set does not seem to have any noticeable effect on the dynamics of the system. Even though not all dynamics are complex, there is no super stable attractor either (the dynamics have a Lyapunov > -0.5). This reflects in the system by the lack of fixed-point response. This can be understood by the presence of the inhibitor, whose role is to avoid those dynamics.

The second experiment tries to find out what percentage of the input space is chaotic. It shows that, as with the "in-supervised" learning, chaos is limited in the system and never fills out the whole state space. Figure 7.13 on page 173 shows this probability after the network learned data sets of varying size. Yet, it appears that unlike the "in-supervised" and "out-supervided" algorithms, this learning does not guarantee chaotic behavior (for lots of data sets the minimum is 0% of chaotic dynamics).

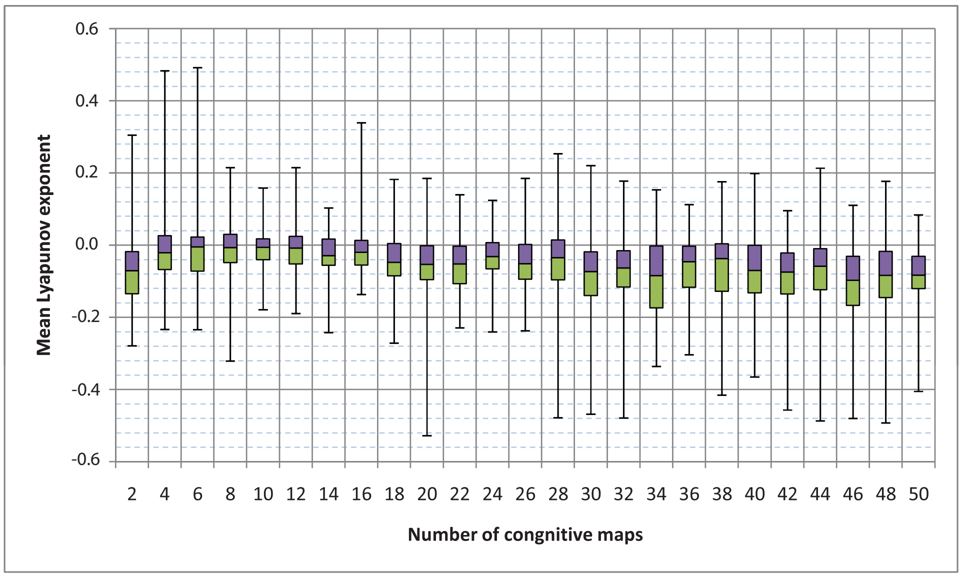


FIGURE 7.12. Dynamics of the system after learning. The mean Lyapunov exponent observed in the system after learning of data sets of increasing size. The result shows that the dynamics of the system averages around 0.0, which indicates weak complex dynamics. However, the variation observed in the measurement indicates that the dynamics can take different forms. The system can become extremely chaotic in some instances, but it is never too stable (the Lyapunov is always greater than -0.5). The main difference observed with this algorithm and the one proposed earlier (Chapter 4 on page 81) is that the increasing data set size does not seem to affect the dynamics of the system, and even a very small data set allows the system to produce complex dynamics.

ε	0.05	It is chosen quite small in order to avoid brutal changes. From experimentation, a smaller value slows the learning process too much, and a larger value induces too strong changes.
γ	0.02	Chosen very small in order to keep the total secondary learning $(\gamma \cdot \varepsilon)$ much smaller than ε
ω	0.2	Avoid weakening process to dampen too much the existing weights
μ	0.95	Allow 95% of the units fed (or feeding) a given cell to be strongly connected $(w_{ij} = 1)$
ν	0.05	It is chosen quite small in order to avoid brutal changes. From experimentation, a smaller value slows the learning process too much, and a larger value induces too strong changes.
$\|A\|$	100	Size of the associative layer.
$\|S\|$	100	Size of the input layer.

TABLE 5. This table recaps all the parameters used for the experiments in Section 4.4 on the previous page with their value.

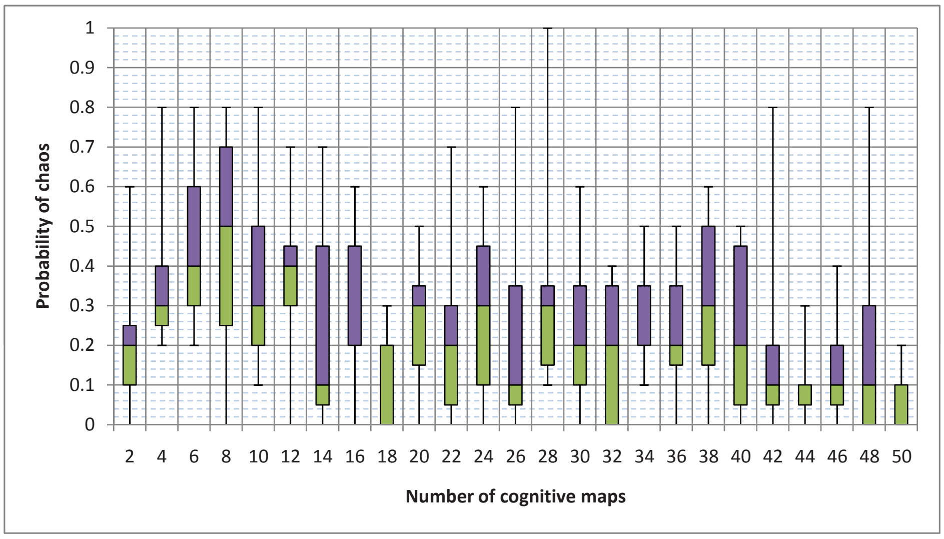


FIGURE 7.13. Probability of chaos after learning. The probability is observed in a system after learning a data set of increasing size. As with the "in-supervised" learning algorithm, the maximum ratio that the chaotic dynamics can cover on the state space seems limited. However, here, the size of the data set does not play a capital role for the system to reach that threshold.

5. The Context Layer

This section will cover the use of contextual information to help the system make successful recalls for ambiguous inputs [Salihoglu et al., 2009b]. Each cognitive map is assumed to have a context. This higher level information is used to distinguish inputs from this cognitive map from other inputs to other cognitive maps. As shown in the previous section, inputs amongst cognitive maps can be strongly correlated, and thus such a process can hopefully help removing some of the ambiguities. The model is composed of the same parts as in the previous sections, but this time the contextual layer is active: this is the full model as reminded in the Figure 7.14 on the following page.

In order to focus on the uses and benefits of this layer the implementation proposed here is kept as simple as possible. The idea is to produce an a priori definition of the context layer (likewise the a priori result obtained for the working memory behavior seen earlier). The size of the context layer is equal to the number of external stimuli stored in the network.

The next step is to define how the connections of the context layer will be determined. In order to fix these connections, it is important to clearly define what is expected from this layer. As the context layer possess one cell for each input of each cognitive map, when the weights are set, each cell should become associated with one of the inputs. Ideally the neurons activated (in the associative layer) with a given input of a cognitive map should activate the corresponding unit in the context layer and reciprocally this unit, when active, should activate the neurons (from the associative layer). To provide a priori contextual information to the system, in these experiments, the five units in the context layer corresponding to the cognitive map can be activated

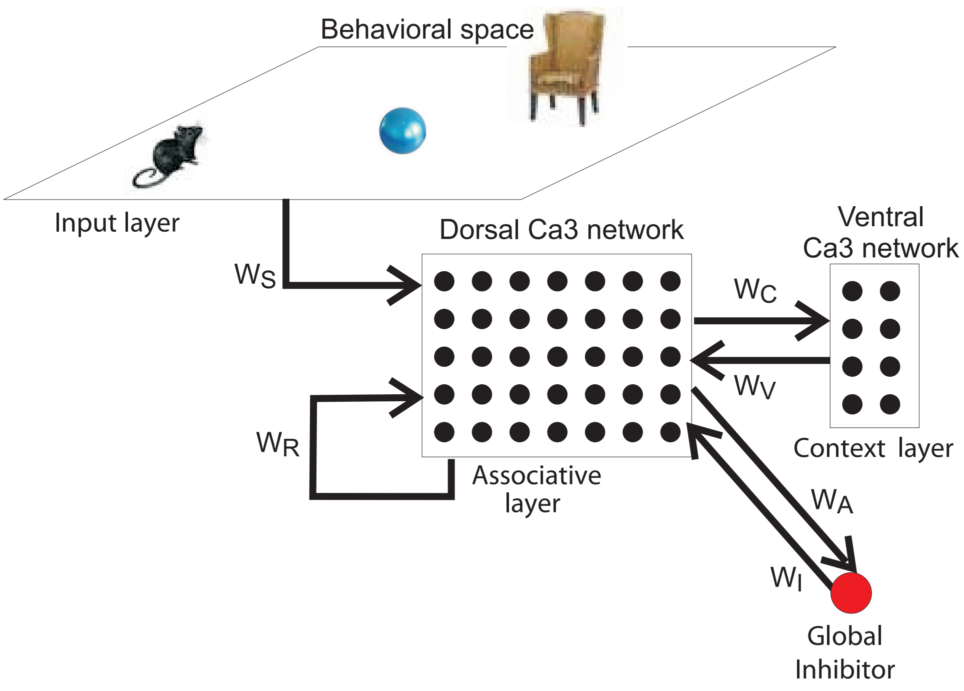


FIGURE 7.14. Architecture of the system. The architecture with all the components.

Connections $\mathbf{W_C}$ and $\mathbf{W_V}$ are set during cell assemblies formation. When a stimulus feeds the network, the back and forth connections between the context cell corresponding to the stimulus and the formed cell assembly are set to 1. This is the easiest way to associate the units, even if this is not very likely from a biological point of view. Nevertheless, this procedure could result from a Hebbian process somehow similar to the associative layer, only working with higher level information.

The basic idea behind a Hebbian solution for the context layer is to manage the contextual layer similarly to the associative layer. This layer will be presented with its own information to learn. This information represents the context (and thus is the same for all inputs in a cognitive map). As with the inputs, it can be assumed that those inputs are pre-formated in a previous layer (such as the dentate gyrus). They can be original information or extracted from the actual stimulus. The inputs can be learned with a Hebbian process and consolidated with the retro-axonal process. The only missing component would be a slower scale Hebbian learning (which can run at similar time scale with the retro-axonal learning). The goal of this last learning would be to learn the weights from and to the context layer from the associative layer ($\mathbf{W_C}$ and $\mathbf{W_V}$).

5.1. Results for Contextual and ambiguous inputs. This section first shows how contextual information is correctly retrieved by the context layer cells. For this first set, two cognitive maps composed of 5 stimuli were encoded, which means that the context layer had to be composed of 10 cells (see section 4.1 on page 163). Figure 7.15 on the facing page shows the normalized activity of the cells associated with the first map (the first five cells) and with the second map (the last five cells), when external stimuli from both maps are successively presented to the network. As expected, when an input from a cognitive map feeds the system the cells of the context layer corresponding to that cognitive map are the most active ones. This shows that the context layer correctly identifies the incoming inputs' context.

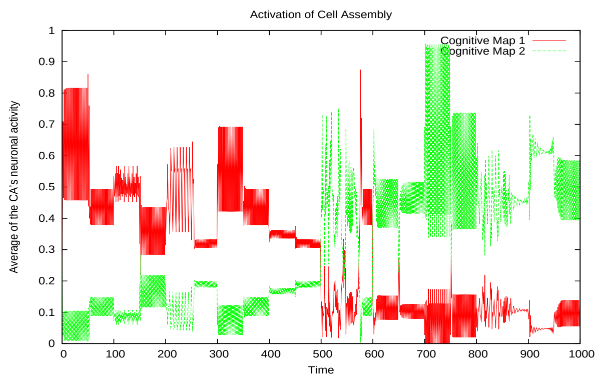


FIGURE 7.15. Normalized activity for units belonging to a same cognitive map. All external stimuli were sequentially presented during 50 time steps, following a 50 time steps period of spontaneous activity. This activity reflects correctly to which cognitive map belongs the cell assembly activity stimulated by the external stimulus.

Figure 7.16 on the next page shows how this contextual information can play an important role to remove the ambiguity in the input layer. To achieve this, the network is fed with external information, which belong equally to a stimulus from the first and the second cognitive maps. Here, the first 4 units of the input layer, belonging to the first stimulus of both maps, were activated (Fig. 7.7 on page 166 shows that these units are equally connected to 'CA 1-CM 1' and 'CA 1-CM 2'). The average activity of these two cell assemblies is monitored as shown by Fig. 7.16 on the next page.

When the context is not predefined, depending on prior spontaneous activity, the network's dynamics settle randomly in one of the CAs and define a context, which then reinforces the working memory feature when the external stimulus is stopped. At time step 200, context cells associated with the first cognitive map were transiently activated. As a result, the network's dynamics settle to the CA 'CA 1-CM 1'. At time step 400, during spontaneous activity, context cells associated with the second cognitive map are activated instead (which could simulate a 'change of mind'). As a result, the network's dynamics settle to the CA associated with the second cognitive map, 'CA 1-CM 2'; and this remains when the stimulus appears again at time 500. As it appears, contextual information can be an useful addition to this model to help remove ambiguity and improve the memory's recall capability.

The next test shows the improvement that can be observed on the recall process when the contextual information is provided to the system alongside the noisy input. Figure 7.17 on the following page shows the benefit from the contextual layer. To do this, the system has to learn 2 cognitive maps. For each learning, the system is fed with increasingly noisy inputs (100 inputs per noise level) and the number of successful recalls is computed. The percentage of successful recalls is computed with or without contextual information. The figure shows the difference observed between the two processes. It is clear that the contextual information provides a boost, but only when the noise level becomes strong enough. This seems natural

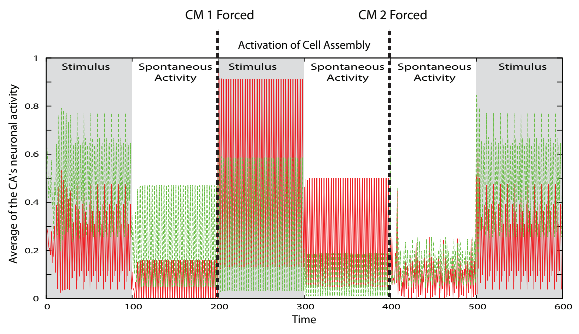


FIGURE 7.16. Reactivation of CA using contextual information. The average activity for two CAs belonging to two different cognitive maps but associated with external stimuli sharing common features (here 4 cells). Here, the first 4 cells of the input layer, belonging both to the first pattern of the cognitive map one and two, are activated at three successive periods: [0 100], [200 300] and [500 600]. Context cells are first set to null. At time 200, context cells associated with the first cognitive map are transiently forced. At time 400, context cells associated with the second cognitive map are transiently forced instead.

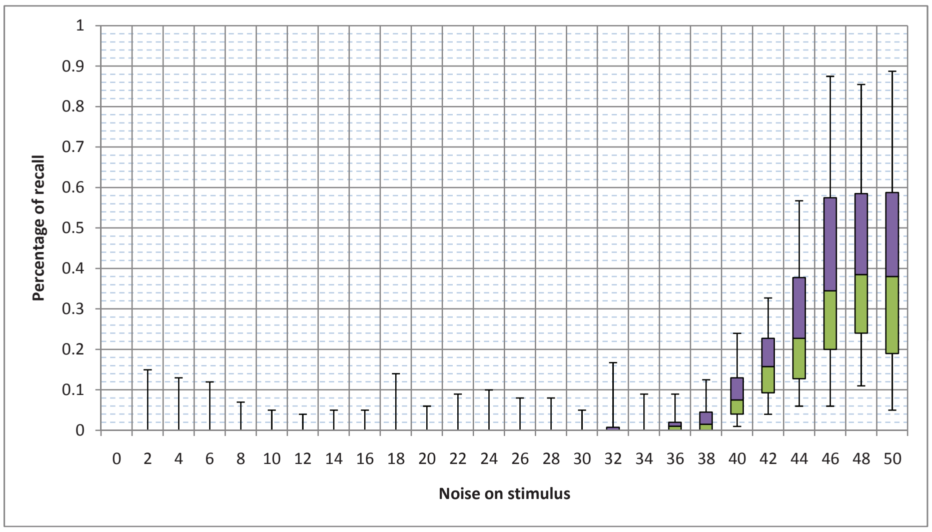


FIGURE 7.17. Benefits of the context layer. Analyzed when the system receives a stimulus with increasing noise. To compute this difference, for each noise level, 100 random stimulus are provided. The output of the system is tested, then it is tested again but this time the contextual information is also provided alongside a noisy input. Then the number of successful recalls for each case is computed and this plots show the increase observed through the contextual information. It appears that the contextual layer becomes useful when noise is very strong. In this experiment, the system had to learn 2 cognitive maps.

since the system already shows a very good recall capability without contextual information (see Figure 7.11 on page 170) when the noise is not too high.

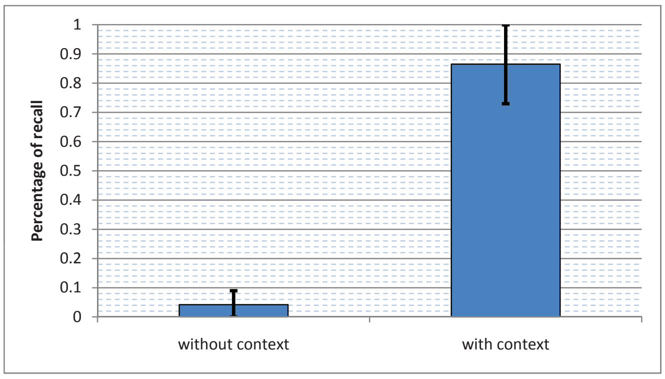


FIGURE 7.18. Benefits of the context layer for ambiguous recall. Shows the percentage of recalls with and without contextual information when the system receive ambiguous stimulus. Here the ambiguous stimulus is obtained by only feeding the neurons that are shared between two stimuli. Such a case is impossible to solve without any further information, since by definition the noisy stimulus is ambiguous and cannot be associated in one way or the other. The benefit from the contextual information is obvious since it completely removes the ambiguity.

The last experiment shows where the contextual information can become mandatory. Indeed, if the system is fed with an ambiguous stimulus, it will need additional information in order to deduce anything. For example, if noise makes it so that the stimulus is only composed of the shared units between two learned stimuli, there is no possible way to know which of these stimuli was the original one. Figure 7.18 shows the percentage of successful recalls when such an ambiguous stimulus feeds the system. In this particular setup the recall boost is naturally impressive.

The main problem with the contextual layer is the proposed implementation. This solution unfortunately does not scale well; it can provide some small boost to the system but fails to provide any convincing results (not shown here). It is easy to understand why. The cell assemblies become smaller as the data sets increase because of the limited possibilities. They also become strongly correlated. Those reasons matter because the contextual layer helps when the system is subject to strong noise. Yet, when learning a large data set, strong noise can distort the input to look like another learned input. The problem with this comes from the transient nature of the contextual information. At each time step the current behavior of the system overrides the contextual information and uses whatever was provided as contextual information for its next step. On the other hand, the input never changes. This means that, at first, the contextual information and the input both send strong opposing informations, but the context layer will change through time and thus, unless the system had quickly converged to a good attractor, the contextual information will get distorted. This means that the input takes the lead and directs the system to the corresponding attractor, which will feedback the context layer with its own contextual information, and after that the context layer will help maintain this attractor.

This suggests that the contextual layer cannot be simply deduced with the simple relation suggested here. To this end, it needs to become a recurrent layer (similar to the associative layer) and needs to learn the contextual information in a similar way to the associative layer. Finally, a slow scale learning algorithm needs to associate the cell assemblies from the associative and context layers. Nevertheless, this does not invalidate any result provided here and these results clearly suggest that the contextual information is important and sometimes even mandatory for a memory-like model.

ε	0.05	It is chosen quite small in order to avoid brutal changes. From experimentation, a smaller value slows the learning process too much, and a larger value induces too strong changes.
γ	0.02	Chosen very small in order to keep the total secondary learning $(\gamma \cdot \varepsilon)$ much smaller than ε
ω	0.2	Avoid weakening process to dampen too much the existing weights
μ	0.95	Allow 95% of the units fed (or feeding) a given cell to be strongly connected $(w_{ij} = 1)$
ν	0.05	It is chosen quite small in order to avoid brutal changes. From experimentation, a smaller value slows the learning process too much, and a larger value induces too strong changes.
$\lvert A\rvert$	100	Size of the associative layer.
$\lvert S\rvert$	100	Size of the input layer.

Table 6. This table recaps all the parameters used for the experiments in Section 5.1 on page 174 with their value.

6. Conclusion

Based on the observations in the previous chapters, recurrent neural networks can provide significant result as memory models. Furthermore, the last chapter has shown that even without a proper learning mechanism, careful a priori weight manipulation can already lead those networks to perform like a valid working memory model. Since it appears that cell assemblies are good candidates as a substrate for information and provide convincing results in the model presented here, the next step obviously was to introduce a learning algorithm that can create those cell assemblies in order to learn the information presented to the system.

In order to mimic a brain-like memory, the model needs to learn the information in a particular way. First, there can be no supervision since it seems not very likely to have an external supervision dictating the brain exactly how it should encode an information. Secondly, the process needs to be autonomous and online. In other words, the system is always learning inputs that it sees. Again this is very reminiscent of the brain, for which the learning can be improved by mechanisms such as rehearsal or attention but is never shut down.

To accommodate those constraints, a simple Hebbian learning has been proposed. This learning tries to create a cell assembly in response to a given stimulus. Depending on the input, some cells will have a Hebbian or anti-Hebbian learning applied to them. Only a subset of the cells are treated on each time step, in order to keep the algorithm less affected by the total size of the system. This algorithm has shown its ability to create cell assemblies, yet proved to have a strong limitation in its capacity to cluster the system in response to the stimuli.

To compensate this an auxiliary algorithm has been proposed. This algorithm would run in a slower time scale and help the Hebbian learning by clustering the stimuli, and by changing the connection the input layer has with the associative layer. This process is based on the retroaxonal hypothesis, which states that the retroaxonal signals present in the brain may be responsible for the stabilization of the newly formed cell assemblies. This hypothesis is gaining a lot of popularity and seems to fit exactly the holes left by the Hebbian learning proposed here.

Once both algorithms have been put together, the performance of the model has been analyzed. First, the simple case proposed in the previous chapter with a priori weight is reproduced here, but this time with learning. The model is able to reproduce the same results. Next the capacity of the system is tested. The system shows that when the number of inputs increases (from 10 inputs) it forgets some of them. However, the system is always able to associate a cell assembly for most of the inputs. Additionally, the system does not seem to show any sign of catastrophic forgetting and old information are naturally erased by new ones. In any case, rehearsal seems to be a simple way to keep information stored, very much like it happens in the human brain's working memory.

Of course, as with any memory, learning large data set is very good but it is useless without noise tolerance. Experiments that have been conducted show that the system performs very well even for very noisy inputs. Those results need to be taken with care however because they are affected by two important facts. First, with the increasing number of randomly generated inputs the probability to have highly correlated inputs rises, and thus at some point a very noisy input can be also obtained with a slight noise applied to another input. This will be interpreted by the system as an occurrence of the second input and thus recorded as a miss in the experiment while the reaction from the system is normal: it reacts with the closest answer. Secondly, the cell assemblies are detected with an heuristic and there is no way to be sure of the result. There is no theoretical help there either, so some tests may seem to fail, where in fact the reason behind it is not the system's performance but the performance of the cell assembly detection algorithm.

As with the previous algorithms, the system's dynamics have also been analyzed, and again the results concord with the hypotheses that have been proposed. Complex dynamics are a part of this model and are brought by the learning mechanism. Like the "in-supervised" algorithm, the dynamics observed here are weak chaos and the probability to find such chaos seems to be limited only to some portion of the states space and does not pollute the whole space. But, unlike the "in-supervised" learning algorithm, here the system produces complex dynamics independently of the data set's size.

Finally, the context layer of the model has been implemented and tested. Here the goal was to show the possibilities of such a layer and how it can help the current model. The goal of this layer is to provide additional information to the system concerning the input, in order to help it distinguish two inputs from two different cognitive maps. Even with a simple a priori definition of the context layer and a simple weight definition between this layer and the associative layer, the system is able to make good use of this layer and sees its performance improved. The results indicate that the contextual information helps removing some of the ambiguity for very noisy inputs. Furthermore, when tested with completely ambiguous stimuli[7] the system without contextual information is unable to make a successful recall while contextual information brings the successful recall rate over 85%.

In conclusion, this chapter introduced a new online unsupervised learning algorithm as a process to create cell assemblies in a working memory in an autonomous

[7]Those stimuli are noisy enough to be good candidates for two inputs and there is no way to know which input is the correct one without additional information.

way. This algorithm is unique since usual working memory model always work from a priori definition of the cell assemblies [Salihoglu et al., 2009b]. In addition, this chapter has taken into consideration the 'retroaxonal' hypothesis and proposed its successful implementation as a process for the stabilization of the cell assemblies formation [Salihoglu et al., 2009b]. Finally, a contextual layer has been proposed to improve this model and has shown some preliminary but conclusive results [Salihoglu et al., 2009b].

CHAPTER 8

Discussion

The goal of this thesis was to produce a brain-like memory model that can boost the memory capacity of Hopfield-like neural networks, based on two biological facts: first, brains are much more than single point attractors. Accordingly, here chaotic background (or spontaneous) dynamics were promoted as much as possible. Second, the memories should reflect internal brain representations and therefore should emerge during a learning process.

During this thesis, a brain-like memory model based on recurrent neural networks and biologically plausible learning algorithms were built. First, a general rate firing model has been introduced. It has been validated from a biological point of view and shown to be computationally efficient (see Chapter 3). To encode memories, a very simple architecture was proposed. The system is composed of an input layer where the external stimuli are presented. This layer has a feed forward connection with the main layer (called the associative layer) and is a recurrent neural network. This main layer can also send feedbacks and is regulated through a global inhibition unit whose role is to keep the system active, within specific bounds. Finally, a context layer feeds and is fed by the associative layer in order to give the system additional information; this allows to remove ambiguities using contextual information. This model is used through all the thesis, even though for some experiments, certain parts of the system are shut down to work with the simplest system which can achieve a given objective. First, a simple input layer was connected to the associative recurrent layer (Chapter 4, 5). Secondly, a global inhibitor was introduced to bound the activities of the system (Chapter 6) and finally, a contextual information layer was exploited (Chapter 7).

The first part of the thesis (Chapter 4, 5) uses the simplest version of this model (an input layer and an associative layer only). Based on observations of Molter and Bersini [2003a,b], it is clear that complex dynamics play an important role for those networks. Following that, learning procedures were proposed to map information into the internal dynamics of the associative layer. This idea has been inspired by two major works: first, Hopfield [1982] showed that a Hebbian prescription rule can be used to encode information into fixed-point attractors of a recurrent neural network. Yet, this model had poor results and severe limitations [Amit et al., 1985]. Secondly, Skarda and Freeman [1987] observed that the brain shows a strong presence of chaos. More specifically, they have shown that natural attentive waiting states correspond to chaotic dynamics, and that the presentation of a known stimulus leads, through bifurcations, to almost cyclic dynamics. The model proposed here takes both those results into consideration: it has been improved to go beyond the limitations of the classical Hopfield [1982] network by the use of external stimuli and to be more in line with the observations of Skarda and Freeman [1987] by the use of cyclic attractors instead of fixed-point attractors. These two modifications have been proven to boost the network's encoding capacity in comparison to the classical Hopfield model (see Chapter 4 and 5).

The next evolution was obviously to use iterative learning algorithms. They are slower to learn a given data set, but they can be configured to learn cyclic information. The first supervised algorithm investigated was the gradient-based backpropagation through time learning algorithm (Chapter 4.2.1 on page 84). It had a poor capability and ended up being very time consuming (roughly on par with a random search). The biggest problem with this algorithm came from the fact that the gradient descent-based approach is often stopped by the numerous bifurcations encountered during the descent. As a consequence, learned networks need to be entirely stable, which in turn leads to a weak number of potential attractors. This can be explained by the fact that high encoding capacity networks have complex internal dynamics incompatible with smooth weight variations. This was not the only flaw of this algorithm: it is also very biologically unlikely as it relies on global information.

Based on those results, the second supervised algorithm (the "out-supervised") tested here followed a quasi-unanimous view shared by neural network researchers: to base the learning of the synaptic matrix on a local Hebbian mechanism. For this purpose, an iterative supervised Hebbian algorithm was created and implemented (Chapter 4, 5) fulfilling the following needs: to encode cyclic information by relying not only on the internal state but also on the external stimuli. Relying on an iterative algorithm has proven to boost the storing capacity compared to the classical Hebbian rule used by Hopfield. However, these results are not new. Though Hebbian learning is an unsupervised local learning rule, the algorithm was still supervised in the sense that Hebbian learning was used to map external stimuli to a priori specified cyclic attractors. On the other hand, the addition of the external stimuli enhances the performance of such an iterative Hebbian learning. This comes from the fact that modifying the stimuli mainly results in a change of the entire internal dynamics, leading to an enlargement of the set of attractors, and potential "memory bags". Those enhancements lead to an increase of the storing capacity and an improvement of the robustness to noise.

While restraining this algorithm to learn fixed-point attractors did not bring any impressive result when the information was encoded into externally specified cyclic attractors, a great improvement of the system's capacity and noise tolerance has been observed. In addition, the background dynamics seemed related to the size of the learned data set. The more information stored, the more chaos becomes unavoidable as the background dynamical regime of the net. In fact, the background chaos spreads widely and adopts a very unstructured shape similar to white noise. In the end, the network is no longer able to manage anything and turns out to be fully and strongly chaotic. By diminishing the amount of learned data, chaotic dynamics show more structure, revealing the presence of nearby stable attractors. Ambiguous inputs lead to ambiguous dynamics [Kelso et al., 1995].

Even though the results obtained using the iterative "out-supervised" Hebbian learning algorithm were not bad, they remained far below the potential expected from these powerful connectionists networks, especially in the light of what can be observed with random small recurrent neural networks [Molter and Bersini, 2003a,b]. Enforcing the cyclic attractors seemed to be too constraining; so the second iterative Hebbian learning that has been proposed (Chapter 4, 5) tries to lighten those constraints. Here, the network has to learn to react to an external stimulus by cycling through a sequence which is not specified a priori, leaving the semantics of the information unprescribed until the learning occurs. This new algorithm (the "in-supervised") already seems to be an improvement, as it is more likely that the exact nature of an encoded information is specific to the brain and not specified a priori from the environment. The attractor associated with the external stimulus is first self-selected by the network, then slightly adapted (by a supervised procedure) in order to make it original and easily identifiable. In this view, the network generates its own relevant information through a self-organized dynamical process. This perspective remains in line with a very old philosophical conviction called constructivism and was modernized in neural network terms by several authors [amongst others Erdi, 1996; Tsuda, 2001; Varela et al., 1991]. In addition, this perspective stays in line with the dynamical hypothesis. The coding scheme is no longer located at the neuron level (i.e. each neuron having its predefined meaning); but instead the interpretation is cast at the dynamical level.

When compared with the first version, huge improvements were seen. The network can learn more, is more tolerant to noise and the learning is faster. In addition, since the algorithm is based on a mechanism of trials–errors–adaptations, larger basins of attractions are naturally obtained for the learned data and thus it does not require extra precaution to ensure noise resistance. The dynamics observed in the system after learning with the "in-supervised" algorithm are very different as well. While the "out-supervised" learning is a road to chaos where the background chaos spreads widely and adopts a very unstructured shape similar to white noise, with the "in-supervised" algorithm the system is much more structured in the obtained chaos. It is still possible to observe the traces of the learned attractors in the chaotic regime. This complex, but still very informative regime, is referred to as the "frustrated chaos". This chaos is characterized by unpredictable itinerancy among nearby learned limit cycle attractors [Bersini, 1998]. Tests showed that when shifting the external stimulus from one learned external stimulus to another one, the traces of the learned attractors are frequently observed in the frustrated regime, giving this chaos a strong and informative structure. This kind of structure and "informative" chaos led several authors to conclude that it could be used to process meaningful information [Sinha and Ditto, 1999; Tsuda, 2001]. This would be in-line with experimental neurophysiological data suggesting that information is carried in the brain through low-dimensional chaos [Rodriguez et al., 1999; Skarda and Freeman, 1990b].

Finally, symbolic investigations have been performed on the spatio-temporal attractors obtained when the network is in a random state and when it is fed by random or noisy stimuli. Different types of attractors can be observed in the output. Spurious data have been defined as attractors having the same period as the learned data but which are still different from all of them. This follows the intuitive idea that the other kinds of attractors (like chaotic attractors) are easily recognizable at a glance and thus are less likely to mislead an observer. In the case of spurious data, it is impossible to know if the observed attractors hold an useful information without comparing it with all the learned data. Networks where the information is coded in fixed point attractors are easily subject to spurious data, which makes them difficult to use. Here, the presence of chaos seems to play an important positive role by preventing the proliferation of spurious data. The effect of this presence can also be actively exploited. Chaotic outputs being easily identifiable, the system can try to deal with them in order to force the convergence of the system to a stable attractor and avoids wandering.

Many researchers from various fields have demonstrated that in well defined circumstances noise plays a positive role. This phenomenon is well exemplified by the stochastic resonance phenomenon, where noise, added to a sub threshold stimulus, can enhance sensory information. Here, noise appears to play a rather different role, since its role is to perturb the ongoing trajectory of the dynamics to make it pass from one attractor to another. The expected result being to change the trajectory from an undesired attractor to the expected learned attractor associated with the feeding stimulus. In line with biological data [Usher and Feingold, 2000], the results reported here indicate that noise can improve memory retrieval. After mapping stimuli to limit cycle attractors of the network's dynamics, noise greatly enhances the retrieval phase by stabilizing the chaotic trajectories to the expected learned cycle. Results also suggest the importance of finding appropriate internal representations of external stimuli. When stimuli are mapped to fixed-point attractors, no chaotic dynamics appear and accordingly noise does not improve the retrieval phase; only spurious information shows up. When mapping stimuli to a priori specified limit cycle attractors, uninformative chaotic dynamics show up. This chaos does not reflect the presence of the nearby competing attractors. Noise does not improve the retrieval phase. When the network chooses by itself how to represent the external stimuli, frustrated chaotic dynamics show up. This chaos shows the presence of nearby attractors, and noise can lead to stabilization of the trajectory to a nearby previously learned attractor.

The last part of the thesis focuses on bringing the learning algorithm ever closer to biological evidences. First, even though the weight-adjusting policy is local (the Hebbian rule) in the two previous algorithms, this rule is applied to all the units at each time step. This somehow negates the advantage that locality provides. Even if the system can still be parallelized, because of the local operations of each weight adjustment, it is still very sensitive to scaling, as each unit must be treated. Also, the learning algorithm still uses supervised components, which is not very likely. Finally, both of the proposed algorithms are offline, which also does not seem adequate for a brain-like memory. The last learning algorithm proposed in this thesis tries to encode information into population networks by relying on complex dynamics to possibly surpass limitations of fixed-point attractors. This cell assembly theory has been suggested by Hebb [1949] and is widely accepted.

The first step (Chapter 6) was to test the model using structured cell assemblies (CA) a priori defined by bi-modal synaptic weights. The model showed very good results and seemed fit to use cell assemblies as a substrate for information. The system was able to learn different cell assemblies with a relatively good noise tolerance. Furthermore, it was also able to exhibit working memory features (i.e. the information carried by the external stimulus could be maintained in the neural dynamics). Dynamical results indicate that large overlapped cell assemblies lead to chaotic regime, whereas smaller and/or uncorrelated cell assemblies tend to give a stable system. While the results look similar to the ones obtained with other, but similar, working memory models [e.g: Compte et al., 2000; Molter et al., 2009; Mongillo et al., 2008], two differences have to be mentioned. First, while other models use complex units (oscillatory units [Molter et al., 2009] or spiking neurons [Compte et al., 2000; Mongillo et al., 2008], here, to allow fast computations and large scale networks, a rate firing model with simple McCulloch and Pitts neurons and with synchronous discrete time step simulation was used. A second difference of the model presented here is that, instead of relying on fixed-point attractors, it enforces complex attractors in CAs by using random weights. As a consequence, chaotic itinerancies amongst previously stored CAs occur.

The second step was to provide a learning algorithm that could achieve the same results autonomously. This algorithm needed to be online and unsupervised which is why a fast Hebbian/anti-Hebbian rule was chosen. However, this learning didn't produce good results, because without supervision nor constraint the learning picks the simplest solution that fits the problem, which in this case is to create one big cell assembly as a response to any stimulus. This isn't very surprising as it is the only thing that was required from the system.

To solve this problem, a second algorithm is proposed to work with the Hebbian process. Basically, without help, the Hebbian learning cannot create a new cell assembly without hurting the old one. The goal of this second learning is to consolidate the cell assemblies that are needed. To this end, this learning is applied to the feed forward connection between the input layer and the associative layer and clusters those connections. This algorithm is inspired by another biologically plausible process: the "retroaxonal hypothesis". The idea is that strong activity in cells will go backward up to the input cell and consolidate the connection (like a reward). However, observations seem to indicate that this retroaxonal signaling is an order of magnitude slower than classic plasticity. In that regard, here the retroaxonal learning is applied much more sparsely than the Hebbian process which is applied at each time step. The slow time scale of the second procedure enables to grasp statistical features of the network's complex dynamics. As a result, input neurons do not get instantaneous supervision from their output as in the classical back propagation algorithm [Rumelhart et al., 1986a,b; Werbos, 1974], neither do they get an instantaneous unsupervised output control, as in the adaptive resonance theory [Grossberg, 1993], but rather a slower and unsupervised 'control' signal. Finally, those algorithms work by only adjusting the weights for a subset of the associative layer and thus scale more easily with the network's size.

Results indicate that when the two learnings are put together the system is able to learn large amounts of information and still stays tolerant to noise. The working memory feature is also preserved. This algorithm also provides complex dynamics as an outcome, but this time it seems to be indifferent of the data set's size. As soon as the algorithm kicks in, the system become slightly chaotic.

Finally, a contextual layer is implemented. Again, a very simple procedure is tested in order to validate the concept and with this a priori definition of the contextual layer some results are obtained. With a low amount of noise, the context layer's help is not very noticeable. This was expected, because in low noise conditions, the system is usually already able to recover from noise without the help of a context layer. But when noise becomes very strong or even a completely ambiguous input is given, contextual information can really help. Unfortunately, this a priori implementation is not strong enough to scale with the data set size, but it suggests that the contextual layer has a purpose and can be improved with an appropriate learning process.

In order to produce all the results in this thesis, a huge effort has been put into the development of a platform that allows easy manipulation of neural networks and learning algorithms. This platform is relatively large and has gone through several major revisions over the years. It is, unfortunately, out of the scope of this thesis to go into its details and inner workings. The interested reader can find more information in the Appendix A on page 189 and Appendix B on page 199.

In conclusion, this thesis proposed a model based on recurrent neural networks, and provided different learning algorithms which allow it to perform memory-related tasks. It showed that, even without relying on biologically plausible learning policies, it is possible to extract useful information from those systems, such as the importance of chaos or how complex dynamics can be used to improve a recall system in a memory. The final step was to propose a biologically likely implementation of a working memory with cell assemblies as a substrate for information. Here, again, the importance of complex dynamics was obvious. Additionally, it showed that a single learning algorithm cannot solve the problem: for good results, it is mandatory to have different layers such as an input layer and a context layer, as well as various learning algorithms working at different time scales. Finally, this work showed that it is possible to use a very simple and computationally friendly model, which can still tackle interesting problems.

Moreover, this thesis concretized a possible use of the retroaxonal hypotheses as suggested by Harris [2008]. This may seem anecdotic, but it is crucial since it is one more evidence that simple models can be used to help understand complex biological problems. The next step in this direction would be to provide a solid contextual layer supported by a plausible implementation and learning algorithm. This could help provide some insight on how correlated information are stored in the brain. This work is based on the hypothesis that there is no single global attractor in the system, which seems to be valid. This idea has been worked upon as this thesis was written and, regarding current results, should probably be the source for some publication following this work.

This model is not only useful from a biological perspective: it can be a robust model for solving various engineering problems. In its current state, it is hard to exploit this model for practical purposes, since the outputs are cell assemblies activities. In contrast, auto-associative memories by definition output something meaningful for the user. A first obvious implementation as an engineering tool for this model would therefore be to provide some form of auto-associative memory feature. This work had also been started during this thesis, but was not mature enough to be discussed in details in this document. The idea is to provide the system with an additional layer (the output layer) whose goal is to reconstruct the input in an auto-associative fashion.

This layer would provide two useful features. First obviously, it provides a user-understandable output from the system. Secondly, since it reconstructs the input, this information is also useful to the system itself. When a noisy input feeds the system, even if the system is able to recover from this noise and has a valid output, the input itself stays noisy and cannot be corrected. On the other hand, the output layer, if successful, will have reconstructed the input as it should be without noise. So, if this layer feedbacks the system with this information, it is very helpful. In such a case, the system has two external information: first, the potentially noisy but genuine information through the input layer; secondly, the normally un-noisy but potentially wrong information from the output layer. A careful usage of both information can obviously lead to improved performances. Early test results suggested that it is possible to implement such a layer.

Nevertheless, this is not the only possible positive outcome for this layer. Current work seems to suggest that the connections from the input layer to the associative layer do not need to be all present. This clustering can be observed in the weight matrix (see Figure 7.7 on page 166). This implies that it may be possible to use a very large input/output layer, sparsely connected to the associative layer. Using such a configuration, one could provide an auto-associative memory where the spatial complexity is converted into spatio-temporal complexity, and thus stored into a reasonable size network, allowing the implementation of an auto-associative memory that will work with real size images and not only toy problems.

The second possible engineering implementation of this model is for clustering problems. The system shows natural inclination for such a problematic and seems to be a perfect fit for stream clustering. In this particular case of clustering, the data is either not accessible all at once or too big to be accessed at once, and thus the clustering must be done on the fly, as data are presented. This is exactly what the model does when different stimuli are presented and similar ones end up activating the same cell assembly, which can be seen as a cluster. However, to be usable the model needs some sort of filtering tool to transform the data and provide the system with an easily understandable input. This is not specific to the clustering and was one of the working hypotheses of the model: the input is not the real one and has been transformed by some mechanism. For the clustering problem, this can probably be achieved with some sort of feed forward neural network whose output layer will be the input layer of this model.

All in all, the model proposed here showed very good results but also opened a lot of doors to expand it both in the biological and engineering realms.

APPENDIX A

Neural Development Kit

The neural development kit (NDK) is a C++ library developed during this thesis. This library has gone through some major changes over the years and its last version is briefly presented here. The goal of this library is to provide an easy access to all the tools needed to work with neural networks.

The library was built with several requirements in mind:

- **Efficiency:** running simulations and tests smoothly, and ability to work with any reasonable network size.
- **Usability:** providing all the basic tools to work with neural networks and being easy to use.
- **Extendability:** being easy to upgrade and to add new features on top of existing ones.
- **Genericity:** being able to work with all kinds of units, networks, algorithms, ...

These are general requirements which need to be respected while proposing an implementation of this library. The NDK is dependent on the Boost[1] library for random numbers generation, fast matrices manipulations and abstract programming (such as functors, auto pointers, ...). Together with the choice of the C++ programming language, this provides a sound base to build an efficient library.

1. Requirements

The main features that this library should provide are summed up here:

- **Design networks:** allow the user to easily create, modify and destroy neural networks with any kind of unit, activation function, architecture, ...
- **Apply learning algorithms:** provide some classic learning algorithms alongside the ability to easily extend them or create new ones from scratch.
- **Manipulate parameters:** allows easy access to all the parameters which can affect any part of the system, with the possibility to save them in a file.
- **Ease simulation:** provide an easy and flexible solution for simulating the network. This implementation must, of course, support synchronous and asynchronous simulation. Moreover, it must provide the possibility to simulate the network in a sandbox in order to be able to unroll any change this simulation might have. This is very important, for example, to compute sensibility to initial conditions, since it requires to simulate a network starting in two almost identical initial conditions.

[1]http://www.boost.org/

- **Provide dynamical and statistical analysis tools:** provide the elementary tools to perform dynamical signal analysis and statistics on any part of the system.

2. Implementation

There are lots of tools on the market designed to manipulate neural networks. Unfortunately, none of them satisfies all those criteria. They are often very static, inflexible, and/or inefficient. Additionally, most of them do not provide their source code, which makes it harder to use for scientific purposes, as there is no way to verify what exactly happens below the hood. Others were discarded due to their lack of quality.

Most of those softwares are lacking in the design department (i.e: high level functional programming, object-oriented design, design patterns, ...). Moreover, they are often programmed in an inappropriate language (i.e: Java, Python, C, ...). All these languages have their uses but do not fit the requirements listed earlier in this annex. For instance, Java and Python are inefficient, where C does not bring anything more than C++, yet lacks very strong tools to provide high-level reusable code.

3. Design

This section gives a general view of the library, then gives more details on the important components. Figure A.1 on the next page shows a general view of the class diagram of the NDK's main actors.

There is of course more to it than just these classes but, before continuing it is important to go over those main actors and explain their use, how they fit the requirements listed earlier, and how they interact.

3.1. BrainData.

3.1.1. *Role.* Contains the raw data which describes the networks. This includes the connections' weight matrix and the internal states vector.

3.1.2. *Implementation.* The *BrainData* does not interact with any class itself, but other classes have to interact with it in order to manipulate the representation of the system. To this end, the *BrainData* sets some friendship[2] with the *Unit* and *Module* classes.

This class does not provide any method of any kind. This is done on purpose: the idea here is that the friendship granted by *BrainData* gives access to the raw data, but nothing more. So it is possible to build a *Brain* class based on this, which can have any internal variable or method, and without any of them being accessible[3]. Both *BrainData*'s attributes have protected accessibility, because *BrainData* is built with the idea to be completed with a *Brain* class.

3.2. Brain.

[2]In a C++ sense: a friend class (or function) has access to private members of the class bestowing the friendship.

[3]Friendship in C++ is not transitive through inheritance or aggregation.

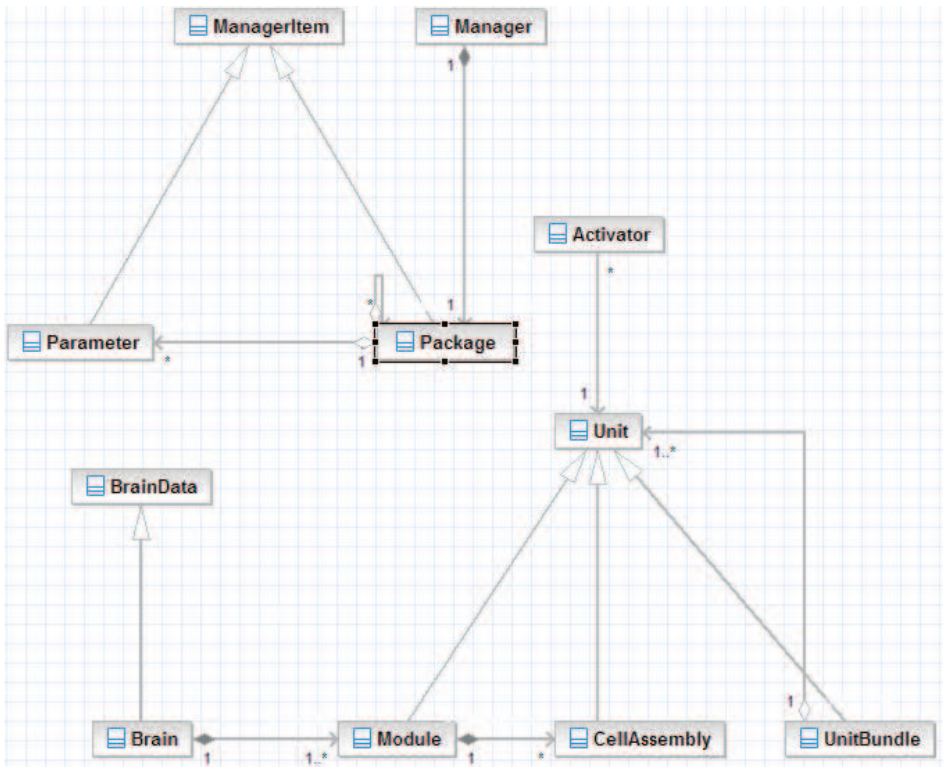


FIGURE A.1. Class diagram of the NDK. This UML class diagram shows the main actors of the neural development kit.

3.2.1. *Role.* Provides an interface for accessing the brain and creating the neural network in it. The *Brain* class represents the system, all the data defining the neural networks, as well as the list of all the existing neural networks. This class is a Singleton[4].

3.2.2. *Implementation.* This class inherits from *BrainData* and defines some additional information of its own. It possesses a collection of Modules[5], and also implements all the necessary functions to manipulate the system. The most important ones are:

- **create:** This method creates a new neural network[6]. This is a factory method [Gamma et al., 1994]: it allows to create a *Module* and thus acts as a constructor for this class. There are three main reasons to provide a factory method: first, it allows the system to have a control on the exact nature of the object that is created. Since this method returns an object, it is possible to provide an extended or more specialized version of *Module* that inherits from *Module*. The best part is that it will still work flawlessly with any legacy code. Secondly, it allows to perform all the necessary tests before creating the object. If those tests were performed in the class' constructor, it could lead to a disaster: if a condition fails, even if the

[4]A singleton is a design pattern introduced by Gamma et al. [1994]. It guarantees that a singleton class can have one and only one instance and that this instance can be accessed globally.

[5]A Module is the implementation of a Neural Network, explained in more detail later.

[6]An instance of the class *Module*.

constructor notifies this with an exception, the object will have already been allocated in memory but will be inconsistent. With the factory method, if the call fails the method never calls the constructor. Finally, it allows the implementation of the class *Module* to hide its constructor. Therefore this method will be the only way to create a neural network. This allows the *Brain* class to track all the created *Modules*, provide some cleanup option and avoid memory leaks. That responsibility has been taken away from the caller, handled by an automated process, yet without taking any performance hit, like with an automated garbage collection.
- **module:** This method gives access to the existing *Modules* in the *Brain.*
- **weights/states:** Two methods give access to the raw data, yet with a subtle protection. The returned object is a new matrix (or vector) which has an access "by reference" to the original matrix (or vector). This means that it is possible to change the value of any weight (or state), but it is not possible to change the matrix (or vector) itself (hence changing its dimension, which is normally dictated by the size of the neural network present in the system, and can only change if some new neural networks get added/removed or new units are inserted/deleted in a neural network.
- **swap/reorder:** Those two methods allow the low-level manipulation of the units' order of appearance in the weight matrix and states vector.

3.3. Unit.

3.3.1. *Role.* The unit describes a group of neurons. It is the base class that will allow to create more specialized groups such as the cell assemblies or the neural network.

3.3.2. *Implementation.* A *Unit* contains a vector of indexes of the neurons that belongs to this group. Using those indexes, it is possible to create a sub-matrix from the original one, which will contain only the weights of the neurons of this group. This attribute has a protected accessibility since the way the indexes are determined depends on the nature of the group of neurons. It is possible from a *Unit* to modify a single value of its weights or states and to have access to the weight matrix (or state vector) defining the *Unit.* Again, like the *Brain*, this matrix (or vector) is a reference to the value of the real matrix (or vector) found in *BrainData.* Here, the returned matrix (or vector) only includes the weight (or states) of the neurons indexed by the *Unit.*

3.4. Module.

3.4.1. *Role.* Defines a neural network composed of a certain number of neurons. It can contain as many cell assemblies as needed and provides all the useful methods to deal with them, in addition to the methods to deal with any group of neurons.

3.4.2. *Implementation.* This class inherits from *Unit* and defines a collection of *Cell Assemblies* (defined in the next section). It receives a vector of indexes to be constructed but, as specified earlier, the constructor is private and this class is friend with the *Brain* class. When the create method of the *Brain* is called, it receives only a size in order to create a neural network of the corresponding size. It resizes the weight matrix and states vector to accommodate the new neurons, and passes the indexes of the newly created neurons to the constructor of *Module.*

In a similar way, *Module* provides the same functionalities to create *Cell Assembly. Module* has a create method and a collection of *Cell Assemblies*, this method receives a list of indexes since a cell assembly is not defined with new neurons;

instead it is just a cluster in an existing *Module*. This is also why the construction is the *Module* class' responsibility and not the *Brain* one.

3.5. Cell Assembly.

3.5.1. *Role.* A *Cell Assembly* is a group of neurons in a given *Module*.

3.5.2. *Implementation.* The *Cell Assembly* inherits from *Unit* as well and defines an attribute of type *Module*, which is a pointer to its parent. This excepted, the cell assembly does not need to provide any feature not common to all groups of neurons, and is thus defined in *Unit*.

3.6. Unit Bundle.

3.6.1. *Role.* This is a composite pattern [Gamma et al., 1994]. Basically, this means that this is a particular *Unit* which contains other *Units*. This allows the user to build any combination of Module, Cell Assembly and even other Unit Bundle into a bundle of units, which can be treated as a single unit composed of different neurons. This allows to look at any group of *Module*/*Cell Assembly* as a single *Unit* and is very useful.

3.6.2. *Implementation.* This class inherits from *Unit* and possesses a container of *Units*. This allows any kind of combination of *Units*. This class defines just two methods that allow to bundle in or out any *Unit*.

3.7. Logger.

3.7.1. *Role.* Maintains the activity of a given list of neurons. This class is used for any measurement of the system's output, without having to manually save the value of the neurons after each simulation.

3.7.2. *Implementation.* This class possesses a reference to the state vector. By default, it has a reference to all the neurons, as it receives the states through the brain, and thus backups for all the neurons. To limit its activity to a single *Unit*, it must, when created, receive as a parameter of its constructor the states it needs to backup at each time step. Theses states are obtained by calling the states method provided in the *Unit* class.

3.8. Function.

3.8.1. *Role.* Provides a set of useful activation functions as well as a prototype to create new ones.

3.8.2. *Implementation.* This is provided as a series of functions defined in a specific namespace: there is no class here as there is no need for one. The system's prototyping is based on a functor[7] from the boost library, which allows the user to define its own function as long as the function receives the correct parameters and returns the correct type. If the function receives more parameters than the prototype, those extra parameters can be binded before creating the functor. In a similar way, if the user wants to create a class because his activation function is very complex and cannot be modeled by a single function, it is also possible. Boost's functors allow the user to create an object of a class, create a functor from

[7]A functor is an object of a class which encapsulates a function. It is a more generic approach to what C/C++ proposes as pointer to function. Basically, it allows to bind a function into it, then manipulate this functor like any other variable, the only difference being that its value is a function. Where a normal variable can be 'read', here the variable can be executed (which triggers the encapsulated function).

a member of that class, and bind the first parameter with the object on which the method should be called[8].

3.9. Activator.

3.9.1. *Role.* This class creates a simulation batch. It allows to simulate a given unit with a given activation function. This class is built either with a reference vector or a copy vector. If it is built with a reference vector, each simulation step will modify the original states from *BrainData*, otherwise this class works locally on a copy. These two cases are respectively called: "in-place activator" and "out-of-place activator".

3.9.2. *Implementation.* This class has a template parameter for the representation of the states vector it will manipulate. Two specialized versions of the template are defined and provide the required functionality ('in-place activator" and "out of place activator"). Yet, as this is a template parameter, it is possible for the user to further specialize this. In addition to that, there is a common interface from which the activator inherits (*ActivatorInterface*), which also allows the user to specialize and create his own activators. This class is constructed using a *Unit* and an activator function[9] or can also receive a third parameter which is a *Logger*. Basically, if this class is created given a *Logger*, it will use it to log each simulation step. This class defines one main function which simulates the given unit with the given activation function and for a certain time.

3.10. Parameter.

3.10.1. *Role.* This class encapsulates the idea of a generic parameter. It is based on a *Manager Item* which defines everything related to parameter configuration, organization and management. Each *Parameter* reflects one configuration parameter of the system and thus allows everyone to access the same value and the modification of this value is reflected to all users.

3.10.2. *Implementation.* Each parameter possesses a name and a description (like all *Manager Item*), in addition to a variant[10] attribute which contains its data.

3.11. Package.

3.11.1. *Role.* A package is a particular *Manager Item* which does not have a data like the *Parameter* but can contains a collection of *Manager Unit.* This is again a composite pattern [Gamma et al., 1994].

3.11.2. *Implementation.* Package has a list of parameters and other packages, and provides operators to access them easily. In addition, this class also allows the creation, destruction and manipulation of any package or parameter.

3.12. Manager.

3.12.1. *Role.* The *Manager* corresponds to a given configuration file: it contains the root package and allows the manipulation of this configuration file, by loading or saving the actual value of the *Packages* and *Parameters*.

[8]A method of a class is nothing more than a function which receives as a first parameter the object itself. For example, *object.func(2)* is interpreted by the compiler as *func(object,2)*.

[9]This is where the functors defined earlier are used

[10]A variant is a class which encapsulates the concept of a variable whose type can be of any kind and change through time. Very much like all the variables in non strongly typed languages.

3.12.2. *Implementation.* This class is created using a factory method present as a static method of the class. This method receives a file name and creates the corresponding manager. If the file already exists, its content is loaded and parsed[11], and the packages and parameters are created accordingly. Any number of configuration files can be manipulated simultaneously.

3.13. Extending the system. Given this structure, it is pretty much possible to do anything. A dynamical analysis tool such as the Lyapunov function is provided as a simple function which works with units and activation functions. A learning algorithm is just a class which, again, works with units and activation functions. The platform already contains various extensions which have been used during this thesis, and also act as proofs of concept of the library. Here is a small example of how to create a network and randomly modify its weight until the Lyapunov exponent observed from a random starting point is greater than 0.0.

[11]The file is saved as an XML file.

```
// define a simulation pair
// composed of a Unit and a Activation function
typedef std::pair<ndk::unit_t ,ndk::sim::function_t> sim_pair_t;

// Get the configuration manager
// loaded from config.xml
ndk::conf::manager_t& mngr=ndk::conf::manager_t::get(``config.
    xml'');
// Set the variable ``out of transient'' in the package ``
    dynamical analysis'' to 20
mngr["dynamical analysis.out of transient"]=20;

// Get the unique instance of the brain
ndk::brain_t& brain=ndk::unique_brain_t();
// Create a modul `mod' of size 3
ndk::module_t& mod=brain.create(3);
// Create a modul `input' of size 3
ndk::module_t& input=brain.create(3);

// `w': the weights matrix from `mod' to `mod' (by reference)
ndk::sub_matrix_t w=mod.weights();
// `i': the weights matrix from `input' to `mod' (by reference)
ndk::sub_matrix_t i=mod.weights(input);
// `ms' is the states vector of `mod' (by reference)
ndk::sub_vector_t ms=mod.states();
// `is' is the states vector of `input' (by reference)
ndk::sub_vector_t is=input.states();

// Create a Lyapunov result struct
ndk::analysis::lyapunov_result_t res;
// Create a vector of pairs containing each unit and its
    activation function.
std::vector<sim_pair_t> simu;
// Add `mod' in this vector with a classic
// synchronous weighted sum, bounded in [0,1]
simu.push_back(std::make_pair(&mod,ndk::sim::
    synchronous_weighted_sum_tanh01));
do
{
  // `w' receives a random uniform distribution in [-1,1]
  w<<=ndk::util::rnd();
  // `i' receives a random uniform distribution in [-1,1]
  i<<=ndk::util::rnd();
  // `ms' receives a random uniform distribution in [0,1]
  ms<<=ndk::util::rnd01();
  // `is' receives a random uniform distribution in [0,1]
  is<<=ndk::util::rnd01();

  // Compute the Lyapunov of `mod'
  // using `simu' as simulation rules.
  res=ndk::an::lyapunov(mod,simu);
// Continue until the mean Lyapunov of `mod' is greater than 0.0
}while (res.mean_lyapunov<0.0);
```

4. Neural Scripting Interface

The NDK has been extended to provide a scripting language. Based on the scripting language "Lua", and using the library Lua and Luabind, the library also provides an exportation of all its core concepts into Lua. This allows the user to use scripts coded in Lua, where the Lua language is extended with concepts introduced in the library using C++. The advantage of such a scripting language is that it is simpler and thus easier to communicate with than the C++ code with whom not lots of people are really familiar. Lua has a syntax which is also simpler and less misleading.

Here is an example of a really simple code in lua :

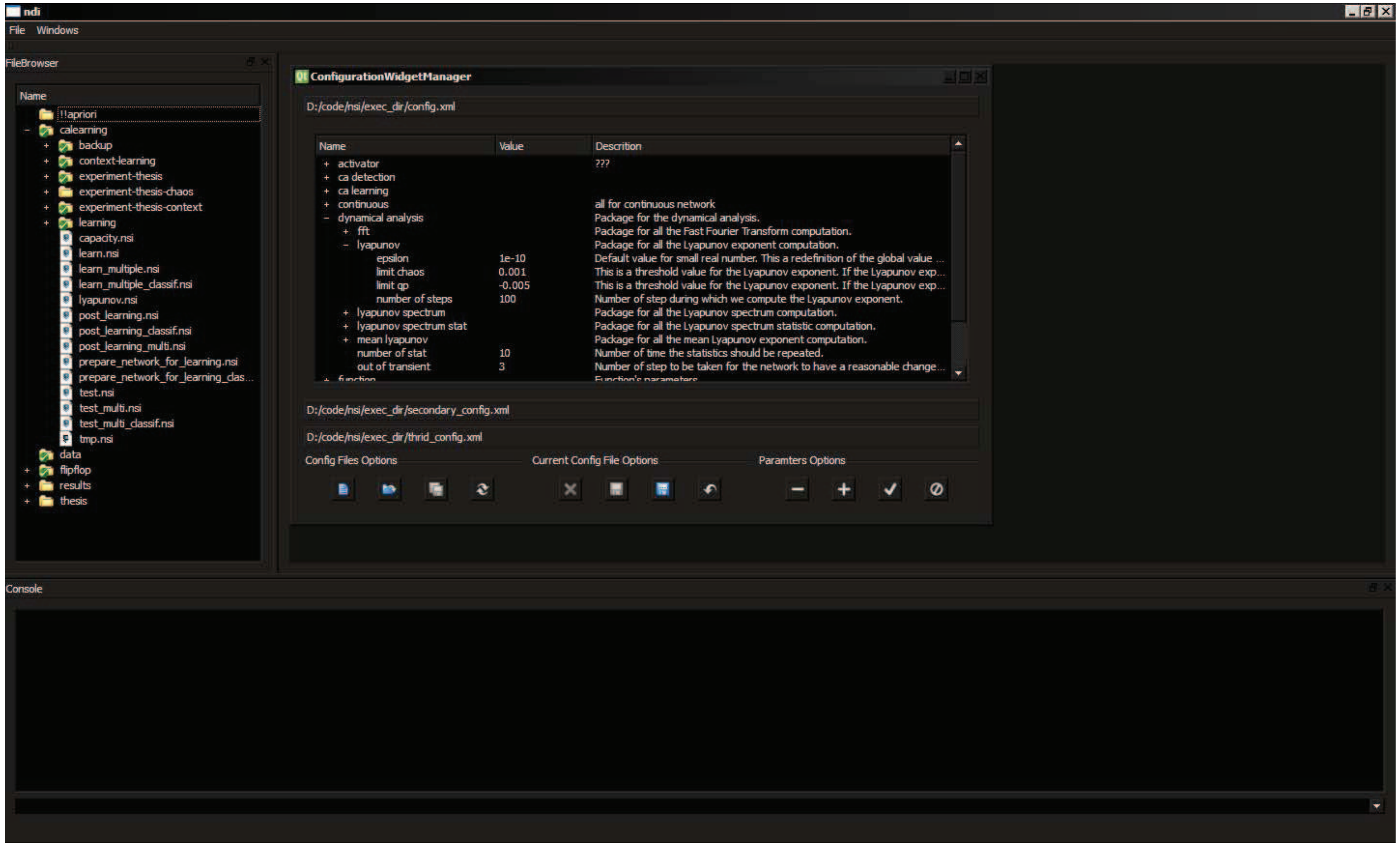
```lua
-- Get the brain.
b=ndk.Brain.get()

-- Create Modules
main=b:create(3)

-- Create Logger
logger=ndk.util.Logger()

-- create Simulator
simu=ndk.sim.InPlaceActivator(main,ndk.sim.
    synchronous_weighted_sum_tanh01())

-- Simulate the network for 10 iterations
simu:simulate(10)
```

APPENDIX B

Neural Development Interface

The neural development interface (NDI) will not be covered in much detail because it mainly consists in the usage of external libraries such as QT. Nevertheless, the idea behind this interface is to go one step further than the scripting interface and provide a Graphical User Interface (GUI). This GUI allows the execution of a script (or script command) through a file browser (or a console respectively). It also exports in Lua different elements specific to the GUI in order to complement the functionality provided by the NDK with Graphic oriented solutions. For instance, it is possible to plot curves, it is also possible to draw the states and weights of any units. This interface also provides a convenient way to modify the configuration parameters, through a configuration manager widget that exposes the tree of packages and parameters. The NDI provides a model/view/control mechanism that easily allows the user to define new widgets that can be used to visualize and control any kind of data.

Several screenshots of this interface in action can be found here. Figure B.1 on the following page shows the configuration manager widget, Figure B.2 on page 201 shows the GUI during a learning process and Figure B.3 on page 202 shows some plots that can be obtain with this interface.



```
Name
    !!apriori
-   calearning
    +   backup
    +   context-learning
    +   experiment-thesis
    +   experiment-thesis-chaos
    +   experiment-thesis-context
    +   learning
        capacity.nsi
        learn.nsi
        learn_multiple.nsi
        learn_multiple_classif.nsi
        lyapunov.nsi
        post_learning.nsi
        post_learning_classif.nsi
        post_learning_multi.nsi
        prepare_network_for_learning.nsi
        prepare_network_for_learning_clas...
        test.nsi
        test_multi.nsi
        test_multi_classif.nsi
        tmp.nsi
    data
+   flipflop
+   results
+   thesis
```

ConfigurationWidgetManager

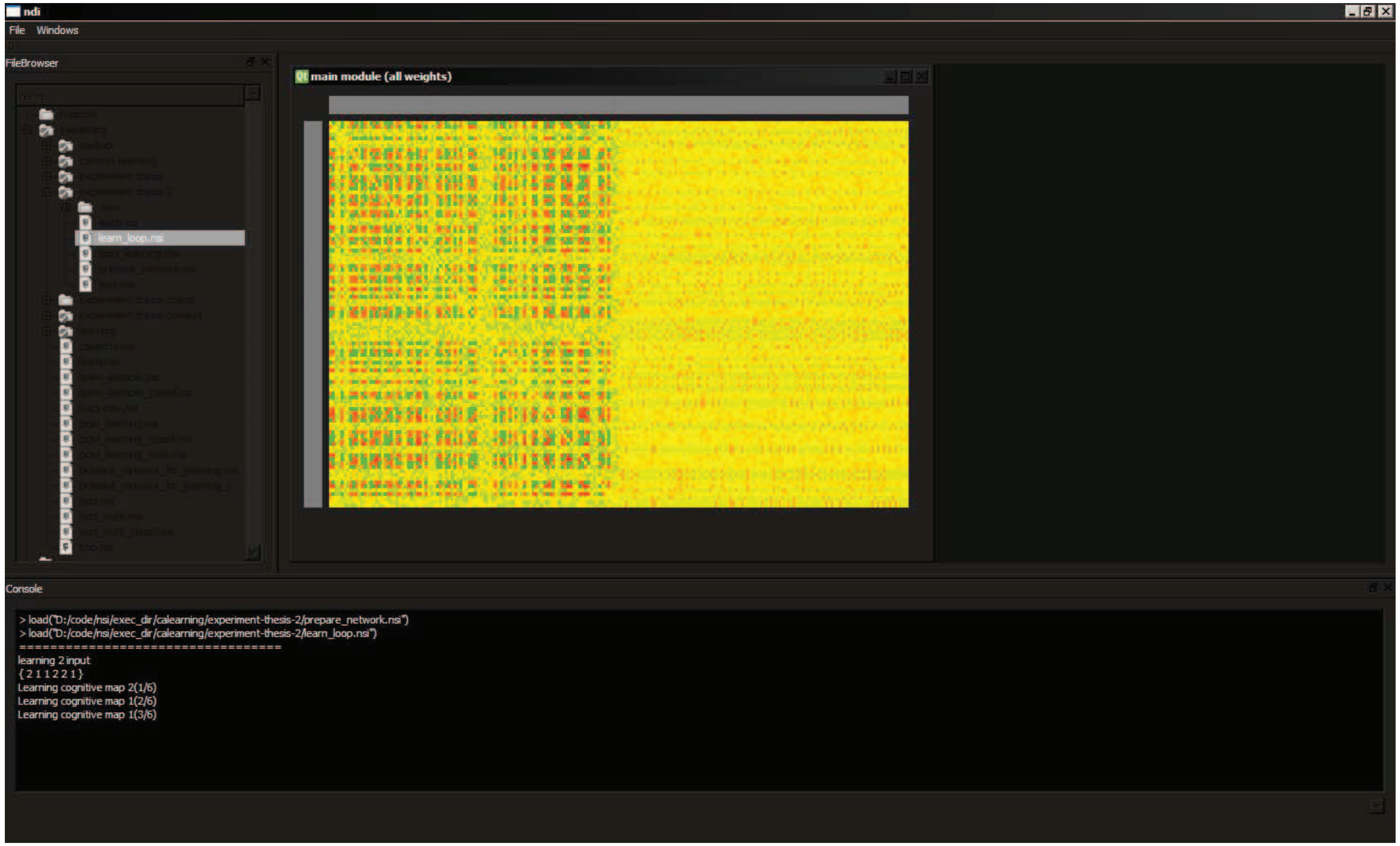
D:/code/nsi/exec_dir/config.xml

Name	Value	Description
+ activator		???
+ ca detection		
+ ca learning		
+ continuous		all for continuous network
− dynamical analysis		Package for the dynamical analysis.
+ fft		Package for all the Fast Fourier Transform computation.
− lyapunov		Package for all the Lyapunov exponent computation.
epsilon	1e-10	Default value for small real number. This a redefinition of the global value ...
limit chaos	0.001	This is a threshold value for the Lyapunov exponent. If the Lyapunov exp...
limit qp	-0.005	This is a threshold value for the Lyapunov exponent. If the Lyapunov exp...
number of steps	100	Number of step during which we compute the Lyapunov exponent.
+ lyapunov spectrum		Package for all the Lyapunov spectrum computation.
+ lyapunov spectrum stat		Package for all the Lyapunov spectrum statistic computation.
+ mean lyapunov		Package for all the mean Lyapunov exponent computation.
number of stat	10	Number of time the statistics should be repeated.
out of transient	3	Number of step to be taken for the network to have a reasonable change...
+ function		Function's parameters

D:/code/nsi/exec_dir/secondary_config.xml

D:/code/nsi/exec_dir/thrid_config.xml

FIGURE B.1. The NDI with the configuration manager widget.



```
> load("D:/code/nsi/exec_dir/calearning/experiment-thesis-2/prepare_network.nsi")
> load("D:/code/nsi/exec_dir/calearning/experiment-thesis-2/learn_loop.nsi")
==================================
learning 2 input
{ 2 1 1 2 2 1 }
Learning cognitive map 2(1/6)
Learning cognitive map 1(2/6)
Learning cognitive map 1(3/6)
```

FIGURE B.2. The NDI during learning.

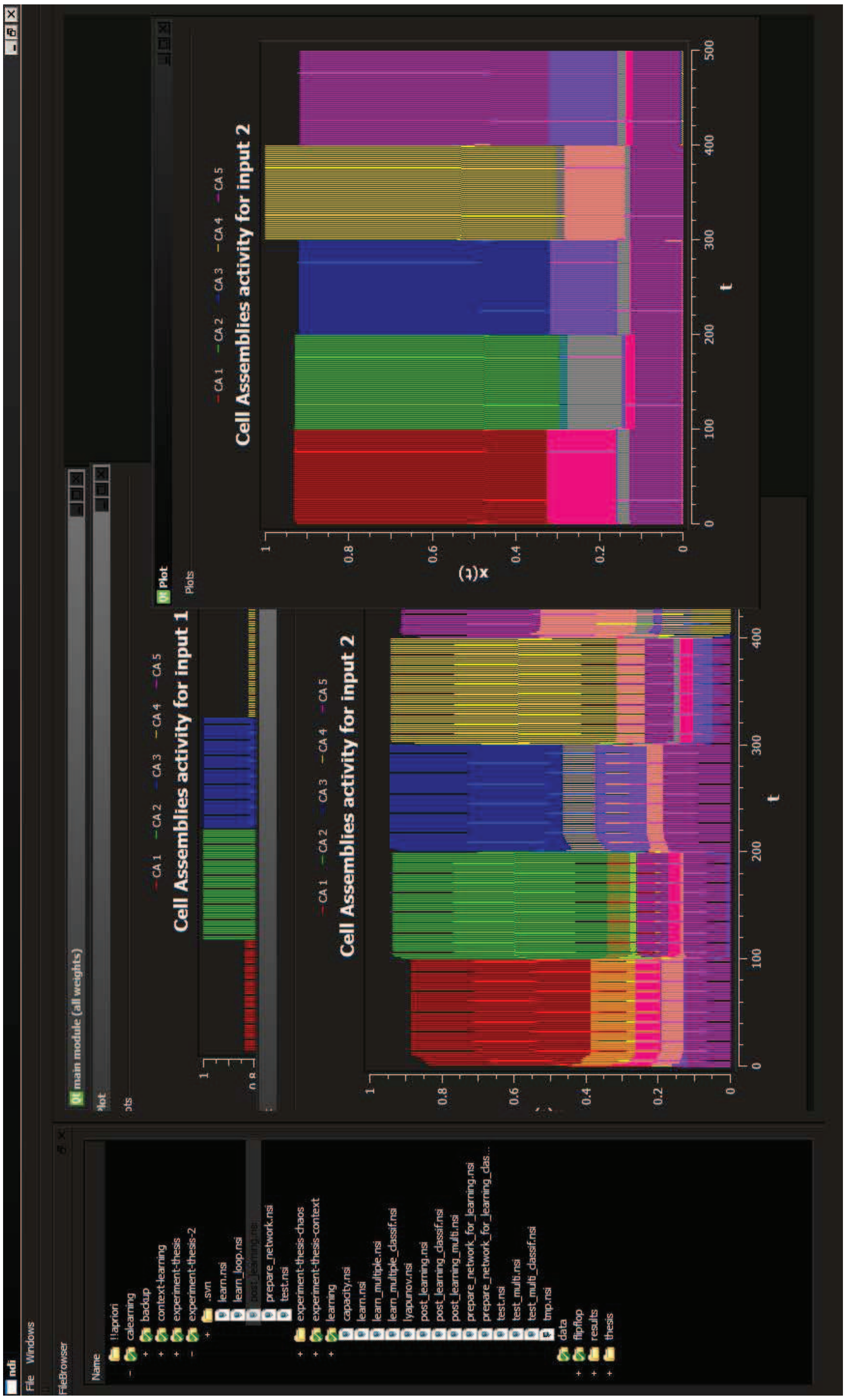


FIGURE B.3. The NDI with some plots.

Bibliography

A

Agnihotri, N. T, Hawkins, R. D, Kandel, E. R, and Kentros, C. The long-term stability of new hippocampal place fields requires new protein synthesis. *Proc Natl Acad Sci U S A*, 101(10):3656–3661, Mar 2004. doi: 10.1073/pnas.0400385101. URL http://dx.doi.org/10.1073/pnas.0400385101. [p. 161]

Albers, D. J, Sprott, J. C, and Dechert, W. D. Routes to Chaos in Neural Networks with random weights. *International Journal of Bifurcation and chaos*, 8:1463–1478, 1998. [p. 55, 62, 112]

Alonso-Nanclares, L, Gonzalez-Soriano, J, Rodriguez, J. R, and DeFelipe, J. Gender differences in human cortical synaptic density. *Proc Natl Acad Sci U S A*, 105(38):14615–14619, Sep 2008. doi: 10.1073/pnas.0803652105. URL http://dx.doi.org/10.1073/pnas.0803652105. [p. 9]

Amaral, D and Lavenex, P. *The Hippocampus Book*, chapter Hippocampal Neuroanatomy, pages 37–114. Oxford University Press, 2006. [p. 29, 30, 31]

Amaral, D. G. A golgi study of cell types in the hilar region of the hippocampus in the rat. *J Comp Neurol*, 182(4 Pt 2):851–914, Dec 1978. [p. 30]

Amari, S. Learning pattern sequences by self-organizing nets of threshold elements. *IEEE Transactions on computers*, C-21:1197–1206, 1972. [p. 81]

Amari, S. Neural theory of association and concept-formation. *Biological Cybernetics*, 26:175–185, 1977. [p. 81]

Amari, S. Field theory of selforganizing neural nets. *IEEE Transactions on Systems, Man, and Cybernetics*, 5(SMC13):741–748, 1983. [p. 15, 60, 64]

Amari, S and Maginu, K. Statistical neurodynamics of associative memory. *Neural Networks*, 1:63–73, 1988. [p. 81]

Amit, D. J. *Modelling Brain Function: the World of Attractor Networks*. Cambridge University Press, New York, 1989. [p. 93, 115, 132]

Amit, D. J. The hebbian paradigm reintegrated: local reverberations as internal representations. *Behavioral Brain Science*, 18:617–657, 1995. [p. 81, 137]

Amit, D. J and Fusi, S. Learning in neural networks with material synapses. *Neural Computation*, 6:957–982, 1994. [p. 81]

Amit, D. J and Mongillo, G. Spike-driven synaptic dynamics generating working memory states. *Neural Computation*, 15:565–596, 2003. [p. 81]

Amit, D. J, Gutfreund, G, and Sompolinsky, H. Spin-glass models of neural networks. *Phys. Rev. A*, 32:1007–1018, 1985. [p. 3, 137, 181]

Amit, D. J, Gutfreund, G, and Sompolinsky, H. Statistical mechanics of neural networks near saturation. *Ann. Phys.*, 173:30–67, 1987. [p. 81, 90]

Ashby, F. G, Ell, S. W, Valentin, V. V, and Casale, M. B. Frost: a distributed neurocomputational model of working memory maintenance. *J Cogn Neurosci*, 17(11):1728–1743, Nov 2005. doi: 10.1162/089892905774589271. URL http://dx.doi.org/10.1162/089892905774589271. [p. 136]

Atkinson, R. C and Shiffrin, R. M. *Human memory: A proposed system and its control processes*, volume 8. London: Academic Press., 1968. [p. 21, 24]

B

Babloyantz, A and Destexhe, A. Lowdimensional chaos in an instance of epilepsy. *Proceedings of National Academy of Sciences*, 83:3513–3517, 1986. [p. 18]

Babloyantz, A and Lourenço, C. Computation with chaos: A paradigm for cortical activity. *Proceedings of National Academy of Sciences*, 91:9027–9031, 1994. [p. 3, 81]

Baddeley, A. D. The influence of acoustic and semantic similarity on long-term memory for word sequences. *Q J Exp Psychol*, 18(4):302–309, Nov 1966. [p. 22]

Baddeley, A. D. *Working memory*. Oxford University Press, New York, 1986. [p. 135]

Baddeley, A. D. *Essentials of human memory*. Psychology Press, 1999. [p. 19]

Baddeley, A. D. The episodic buffer: A new component of working memory? *Trends in Cognitive Science*, 4:417–423, 2000. [p. 25]

Baddeley, A. D and Hitch, G. J. *The psychology of learning and motivation: advances in research and theory*, volume 8, chapter Working Memory, pages 47–89. New York: Academic Press, 1974. [p. 6, 22, 25]

Baddeley, A. D, Thomson, N, and Buchanan, M. Word length and the structure of short term memory. *Journal of Verbal Learning and Verbal Behavior*, 14:575–589, 1975. [p. 20]

Barlow, H. B. Single neurons and sensation: A neuron doctrine for perceptual psychology. *Perception*, 1:371–394, 1972. [p. 15]

Bengio, Y, Simard, P, and Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, March 1994. [p. 97]

Bersini, H. The frustrated and Compositionnal Nature of Chaos in Small Hopfield Networks. *Neural Networks*, 11:1017–1025, 1998. [p. 120, 183]

Bersini, H and Calenbuhr, V. Frustrated chaos in biological networks. *J. theor. Biol*, 177:199–213, 1997. [p. 120]

Bersini, H and Sener, P. The connections between the frustrated chaos and the intermittency chaos in small hopfield networks. *Neural Netwoks*, 15:1197–1204, 2002. [p. 120]

Best, P. J and White, A. M. Placing hippocampal single-unit studies in a historical context. *Hippocampus*, 9(4):346–351, 1999. doi: 3.0.CO;2-3. URL `http://dx.doi.org/3.0.CO;2-3`. [p. 27]

Bi, G and Poo, M. Distributed synaptic modification in neural networks induced by patterned stimulation. *Nature*, 401:792–796, 1999. [p. 5, 92]

Bland, J. *About Gender: Nerves*. 1998. [p. 13]

Bliss, T. V and Lomo, T. Long-lasting potentiation of synaptic transmission in the dentate area of the anaesthetized rabbit following stimulation of the perforant path. *Journal of Physiology*, 232:331–356, July 1973. [p. 14, 81]

Broadbent, N. J, Clark, R. E, Zola, S, and Squire, L. R. *The Neuropsychology of Memory*, chapter The Medial Temporal Lobe and Memory, pages 3–23. Guilford Press, 2002. [p. 28]

Brooks, R. A. A robust layered control system for a mobile robot. *IEEE Transactions on Robotics and Automation*, 2:14–23, 1986. [p. 65]

Brown, G. D. A, Neath, I, and Chater, N. A ratio model of scale-invariant memory and identification. *Psychological Review*, 114:539–576, 2007. [p. 21]

Brunel, N and Wang, X. J. Effects of neuromodulation in a cortical network model of object working memory dominated by recurrent inhibition. *J Comput Neurosci*, 11(1):63–85, 2001. [p. 137]

Brunel, N, Carusi, F, and Fusi, S. Slow stochastic hebbian learning of classes of stimuli in a recurrent neural network. *Network: Computation in Neural Systems*, 9:123–152, 1997. [p. 81]

Burrow, T. The neurodynamics of behavior. a phylobiological foreword. *Philosophy of Science*, 10(4):271–288, 1943. [p. 16]

Buss, R. R, Sun, W, and Oppenheim, R. W. Adaptive roles of programmed cell death during nervous system development. *Annu Rev Neurosci*, 29:1–35, 2006. doi: 10.1146/annurev.neuro.29.051605.112800. URL http://dx.doi.org/10.1146/annurev.neuro.29.051605.112800. [p. 6, 159]

Butcher, J. C. *Numerical methods for ordinary differential equations*. Wiley, 2003. [p. 73]

Buzsáki, G. Theta oscillations in the hippocampus. *Neuron*, 33(3):325–340, Jan 2002. [p. 31]

Buzsáki, G. Hippocampal sharp waves: their origin and significance. *Brain Res.*, 398:242–252, 1986. [p. 31]

Buzsáki, G. A two-stage model of memory trace formation: A role for "noisy" brain states. *Neuroscience*, 31:551–570, 1989. [p. 33]

Buzsáki, G. *Rhythms of the Brain*. Oxford University Press, USA, 2006. [p. 31]

Buzsáki, G, Chen, L. S, and Gage, F. H. Spatial organization of physiological activity in the hippocampal region: relevance to memory formation. *Prog Brain Res*, 83:257–268, 1990. [p. 31]

C

Cantero, J. L, Atienza, M, Stickgold, R, Kahana, M. J, Madsen, J. R, and Kocsis, B. Sleep-dependent theta oscillations in the human hippocampus and neocortex. *J Neurosci*, 23(34):10897–10903, Nov 2003. [p. 33]

Carpenter, G and Grossberg, S. The art of adaptive pattern recognition by a self-organizing neural network. *Computer*, 21:77–88, 1988. [p. 6, 64, 65, 93]

Carpenter, G. A and Grossberg, S. Adaptive resonance theory. In Arbib, M. A, editor, *The Handbook of Brain Theory and Neural Network*. MIT Press, 2003. [p. 64]

Cessac, B. *Systèmes dynamiques de grande taille et mécanique statistique*. Institut Non-Linéaire de Nice, Nice, 2002. Habilitation à diriger les recherches. [p. 44, 60]

Changeux, J. P. *L'homme neuronal*. Fayard, 1985. [p. 13]

Chiu, Y.-C, Algase, D, Whall, A, Liang, J, Liu, H.-C, Lin, K.-N, and Wang, P.-N. Getting lost: directed attention and executive functions in early alzheimer's disease patients. *Dement Geriatr Cogn Disord*, 17(3):174–180, 2004. doi: 10.1159/000076353. URL http://dx.doi.org/10.1159/000076353. [p. 29]

Cohen, I, Navarro, V, Clemenceau, S, Baulac, M, and Miles, R. On the origin of interictal activity in human temporal lobe epilepsy in vitro. *Science*, 298(5597):1418–1421, 2002. [p. 18]

Cohen, J. D, Perlstein, W. M, Braver, T. S, Nystrom, L. E, Noll, D. C, Jonides, J, and Smith, E. E. Temporal dynamics of brain activation during a working memory task. *Nature*, 386(6625):604–608, Apr 1997. doi: 10.1038/386604a0. URL http://dx.doi.org/10.1038/386604a0. [p. 6, 136]

Cohen, M and Grossberg, S. Neural networks and physical systems with emergent computational abilities. *Proc. Natl. Acad. Sci. USA*, 1982. [p. 66]

Cohen, N. J and Eichenbaum, H. *Memory, Amnesia, and the Hippocampal System*. MIT Press, 1993. [p. 28]

Collins, J. J, Chow, C. C, and Imhoff, T. T. Stochastic resonance without tuning. *Nature*, 376:236–238, 1995. [p. 5]

Compte, A, Brunel, N, Goldman-Rakic, P. S, and Wang, X. J. Synaptic mechanisms and network dynamics underlying spatial working memory in a cortical network model. *Cereb Cortex*, 10(9):910–923, Sep 2000. [p. 6, 138, 140, 141, 184]

Conrad, R. Acoustic confusions in immediate memory. *British Journal of Psychology*, 55:75–84, 1964. [p. 20]

Cooke, S. F and Bliss, T. V. P. Plasticity in the human central nervous system. *Brain*, 129(7):1659–1673, 2006. [p. 14]

Cowan, N. *Attention and memory: An integrated framework*. New York: Oxford University, 1995. [p. 25]

Cowan, N. The magical number 4 in short-term memory: a reconsideration of mental storage capacity. *Behav Brain Sci*, 24(1):87–114; discussion 114–85, Feb 2001. [p. 20]

Cowan, N. *Working memory capacity*. New York, NY: Psychology Press, 2005. [p. 25]

Craik, F. I and Lockhart, R. S. Levels of processing: A framework for memory research. *Journal of Verbal Learning & Verbal Behavior*, 11(6):671–684, 1972. [p. 23]

Curtis, C and D'Esposito, M. Persistent activity in the prefrontal cortex during working memory. *Trends Cogn Sci*, 7(9):415–423, Sep 2003. [p. 24]

Cybenko, G. Approximation by superpositions of a sigmoidal function. *Math. Control Signals Systems*, 2:303–314, 1989. [p. 34, 37]

D

Dauce, E. *Adaptation dynamique et apprentissage dans les réseaux de neurones récurrents aléatoires*. PhD thesis, SUPAERO, 2000. [p. 55]

Dauce, E, Quoy, M, Cessac, B, Doyon, B, and Samuelides, M. Self-organization and dynamics reduction in recurrent networks: stimulus presentation and learning. *Neural Networks*, 11:521–533, 1998. [p. 64]

Davelaar, E. J, Goshen-Gottstein, Y, Ashkenazi, A, Haarmann, H. J, and Usher, M. The demise of short-term memory revisited: empirical and computational investigations of recency effects. *Psychol Rev*, 112(1):3–42, Jan 2005. doi: 10.1037/0033-295X.112.1.3. URL `http://dx.doi.org/10.1037/0033-295X.112.1.3`. [p. 21]

Destexhe, A and Contreras, D. Neuronal computations with stochastic network states. *Science*, online, 2006. [p. 5]

Devaney, R. L. *An introduction to chaotic dynamical systems*. Addison Wesley, second edition, 1989. [p. 41, 46]

Diana, R. A, Yonelinas, A. P, and Ranganath, C. Imaging recollection and familiarity in the medial temporal lobe: a three-component model. *Trends Cogn Sci*, 11(9):379–386, Sep 2007. doi: 10.1016/j.tics.2007.08.001. URL `http://dx.doi.org/10.1016/j.tics.2007.08.001`. [p. 28]

Domany, E, van Hemmen, J, and Schulten, K. *Models of Neural Networks*, volume 1. Springer, 2nd edition, 1995. [p. 3, 81]

Doya, K. Bifurcations in the learning of recurrent neural networks. In *Proc. of 1992 IEEE Int. Symposium on Circuits and Systems*, pages 2777–2780, 1992. [p. 97]

Doyon, B, Cessac, B, Quoy, M, and Samuelides, M. Control of the transition to chaos in neural networks with random connectivity. *Int. J. Bifurcation and Chaos*, 3:279–291, 1993. [p. 55, 64]

Doyon, B, Quoy, M, and Samuelides, M. Mean-field equations, bifurcation map and route to chaos in discrete time neural networks. *Physica D*, 74:24–44, 1994. [p. 64]

Du, J.-L and Poo, M.-M. Rapid bdnf-induced retrograde synaptic modification in a developing retinotectal system. *Nature*, 429(6994):878–883, Jun 2004. doi: 10.1038/nature02618. URL `http://dx.doi.org/10.1038/nature02618`. [p. 159]

Durstewitz, D, Seamans, J. K, and Sejnowski, T. J. Neurocomputational models of working memory. *Nat Neurosci*, 3 Suppl:1184–1191, Nov 2000. doi: 10.1038/81460. URL http://dx.doi.org/10.1038/81460. [p. 6, 135, 136]

E

Eckmann, J. P and Ruelle, D. Ergodic theory of chaos and strange attractors. *Reviews of modern physics*, 57(3):617–656, July 1985. [p. 45, 48, 52]

Eichenbaum, H, Otto, T. A, Wible, C. G, and Piper, J. M. *Olfaction: A Model System for Computational Neuroscience*, chapter Building a model of the hippocampus in olfaction and memory. MIT Press, 1991. [p. 27]

Eichenbaum, H, Yonelinas, A. P, and Ranganath, C. The medial temporal lobe and recognition memory. *Annu Rev Neurosci*, 30:123–152, 2007. doi: 10.1146/annurev.neuro.30.051606.094328. URL http://dx.doi.org/10.1146/annurev.neuro.30.051606.094328. [p. 31]

Ekstrom, A. D, Kahana, M. J, Caplan, J. B, Fields, T. A, Isham, E. A, Newman, E. L, and Fried, I. Cellular networks underlying human spatial navigation. *Nature*, 425(6954):184–188, Sep 2003. doi: 10.1038/nature01964. URL http://dx.doi.org/10.1038/nature01964. [p. 28]

Elaydi, S. N. *Discrete Chaos*. Chapman & Hall/CRC, 1999. [p. 47]

Elman, J. L. Distributed representations, simple recurrent networks, and grammatical structure. In *Connectionist Approaches to Language Learning*, pages 91–122. Touretzky, 1991. [p. 15]

Erdi, P. The brain as a hermeneutic device. *Biosystems*, 38:179–189, 1996. [p. 93, 183]

Ericsson, K. A and Kintsch, W. Long-term working memory. *Psychol Rev*, 102(2):211–245, Apr 1995. [p. 25]

Eysenck, M. W and Eysenck, M. C. Effects of processing depth, distinctiveness, and word frequency on retention. *British journal of psychology*, 71(2):263–274, 1980. [p. 24]

F

Feigenbaum, M. J. Universal behavior in nonlinear systems. *Physica D*, 7:16–39, 1983. [p. 46, 55]

Finger, S. *Origins of Neuroscience: A History of Explorations Into Brain Function*. Oxford University Press US, 2001. [p. 27]

Fitzsimonds, R. M, Song, H. J, and Poo, M. M. Propagation of activity-dependent synaptic depression in simple neural networks. *Nature*, 388(6641):439–448, Jul 1997. doi: 10.1038/41267. URL http://dx.doi.org/10.1038/41267. [p. 159]

Forrest, B. M and Wallace, D. J. *Models of Neural networks*, volume 1, chapter Storage capacity and learning in ising-spin neural networks, pages 129–156. Springer, 2nd edition, 1995. [p. 90, 91, 102]

Frank, L. M, Stanley, G. B, and Brown, E. N. Hippocampal plasticity across multiple days of exposure to novel environments. *J Neurosci*, 24(35):7681–7689, Sep 2004. doi: 10.1523/JNEUROSCI.1958-04.2004. URL http://dx.doi.org/10.1523/JNEUROSCI.1958-04.2004. [p. 160]

Franosch, J.-M, Lingenheil, M, , and van Hemmen, J. How a frog can learn what is where in the dark. *Physical Review Letters*, 95:1–4, 2005. [p. 96]

Freeman, W. J. A proposed name for aperiodic brain activity: stochastic chaos. *Neural Networks*, 13:11–13, 2000. [p. 17]

Freeman, W. J. *Biocomputing*, chapter Making sense of brain waves: the most baffling frontier in neuroscience, pages 1–23. Kluwer Academic, 2002. [p. 4, 16]

Freeman, W. J, Chang, H.-J, Burke, B. C, Rose, P. A, and Badler, J. Taming chaos: stabilization of aperiodic attractors by noise. *IEEE Trans. on Circuits and Systems — 1*, 44(10):989+, 1997. URL `citeseer.ist.psu.edu/freeman97taming.html`. [p. 5]

Freud, S. *The Project for a Scientific Psychology*. 1895. [p. 35]

Fusi, S. Hebbian spike-driven synaptic plasticity for learning patterns of mean firing rates. *Biological Cybernetics*, 87:459–470, 2002. [p. 81, 115]

Fuster, J. M. Unit activity in prefrontal cortex during delayed-response performance: neuronal correlates of transient memory. *J Neurophysiol*, 36(1):61–78, Jan 1973. [p. 6, 135]

Fuster, J. M and Alexander, G. E. Neuron activity related to short-term memory. *Science*, 173(997):652–654, Aug 1971. [p. 6, 136]

G

Gamma, E, Helm, R, Johnson, R, and Vlissides, J. M. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1994. [p. 191, 193, 194]

Gardner, E. Maximum storage capacity in neural networks. *Europhys. Lett.*, 4: 481–485, 1987. [p. 81, 82, 90, 91, 132, 137]

Gardner, E and Derrida, B. Three unfinished works on the optimal storage capacity of networks. *J. Physics A: Math. Gen.*, 22:1983–1994, 1989. [p. 81]

Gazzaniga, M. S. *Nature's Mind, The Biological Roots of Thinking, Emotions, Sexuality, Language, and Intelligence*. Basic Books, 1992. [p. 13]

Gehring, W. J. New perspectives on eye development and the evolution of eyes and photoreceptors. *J Hered*, 96(3):171–184, 2005. doi: 10.1093/jhered/esi027. URL `http://dx.doi.org/10.1093/jhered/esi027`. [p. 9]

Gobet, F. Some shortcomings of long-term working memory. *Br J Psychol*, 91 (Pt 4):551–570, Nov 2000. [p. 26]

Goldman-Rakic. *Models of Information Processing in the Basal Ganglia*, chapter Toward a Circuit Model of Working Memory and the Guidance of Coluntary Motor Action, pages 131–148. MIT Press, Cambridge, Massachusetts, 1994. [p. 136]

Gray, C. M, Köning, P, Engel, A. K, and Singer, W. Oscillatory responses in cat visual cortex exhibit intercolumnar synchronization which reflects global stimulus properties. *Nature*, 338:334–337, 1989. [p. 17]

Gray, J. A and McNaughton, N. *The Neuropsychology of Anxiety: An Enquiry into the Functions of the Septo-Hippocampal System*. Oxford University Press, 2000. [p. 27]

Grillner, S and Wallén, P. Cellular bases of a vertebrate locomotor system-steering, intersegmental and segmental co-ordination and sensory control. *Brain Res Brain Res Rev*, 40(1–3):92–106, Oct 2002. [p. 9]

Grossberg, S. *Mathematical approaches to neural networks*, chapter Self-organizing neural networks for stable control of autonomous behavior in a changing world, pages 139–197. Amsterdam : Elsevier Science Publishers, 1993. [p. 6, 185]

Grossberg, S. *Neural Networks and Natural Intelligence*. MIT Press, Cambridge, 1992. [p. 3, 81]

Guida, A and Tardieu, H. Is personalisation a way to operationalise long-term working memory? *Current Psychology Letters: Behaviour, Brain & Cognition*, 15(1):1–15, 2005. [p. 26]

Guida, A, Tardieu, H, and Nicolas, S. The personalisation method applied to a working memory task: evidence of long-term working memory effects. *European Journal of Cognitive Psychology.*, 2008. [p. 26]

Guillot, A and Dauce, E. *Approche dynamique de la cognition artificielle*. Hermès Science, Paris, 2002. [p. 4, 14, 17, 65]

Gutfreund, Y, Zheng, W, , and Knudsen, E. I. Gated visual input to the central auditory system. *Science*, 297:1556–1559, 2002. [p. 96]

H

Hamburger, V. History of the discovery of neuronal death in embryos. *J Neurobiol*, 23(9):1116–1123, Nov 1992. doi: 10.1002/neu.480230904. URL http://dx.doi.org/10.1002/neu.480230904. [p. 6, 159]

Hamburger, V. The history of the discovery of the nerve growth factor. *J Neurobiol*, 24(7):893–897, Jul 1993. doi: 10.1002/neu.480240702. URL http://dx.doi.org/10.1002/neu.480240702. [p. 6, 159]

Hameroff, S and Penrose, R. Orchestrated reduction of quantum coherence in brain microtubules: A model for consciousness? In Hameroff, S. R, Kaszniak, A. W, and Scott, A. C, editors, *Toward a Science of Consciousness - The First Tucson Discussions and Debates*, pages 507–540. Cambridge, MIT Press, 1996. [p. 15]

Harris, K. D. Neural signatures of cell assembly organization. *Nat Rev Neurosci*, 6(5):399–407, May 2005. doi: 10.1038/nrn1669. URL http://dx.doi.org/10.1038/nrn1669. [p. 138]

Harris, K. D. Stability of the fittest: organizing learning through retroaxonal signals. *Trends Neurosci*, 31(3):130–136, Mar 2008. doi: 10.1016/j.tins.2007.12.002. URL http://dx.doi.org/10.1016/j.tins.2007.12.002. [p. 6, 159, 161, 186]

Harter, D and Kozma, R. Chaotic neurodynamics for autonomous agents. *IEEE Trans Neural Netw*, 16(3):565–579, May 2005. doi: 10.1109/TNN.2005.845086. URL http://dx.doi.org/10.1109/TNN.2005.845086. [p. 64]

Hayek, F. A. *The Sensory Order: An Inquiry into the Foundations of Theoretical Psychology*. Routledge & Kegan Paul PLC, 1952. [p. 35]

Haykin, S. *Neural Networks - A Comprehensive Foundation (2nd. ed.)*. Prentice-Hall, Upper Saddle River, NY, 1998. [p. 35, 39, 65]

Hebb, D. *The organization of behavior*. Wiley-Interscience, New York, 1949. [p. 6, 14, 15, 33, 40, 66, 81, 135, 136, 138, 146, 184]

Hippenmeyer, S, Kramer, I, and Arber, S. Control of neuronal phenotype: what targets tell the cell bodies. *Trends Neurosci*, 27(8):482–488, Aug 2004. doi: 10.1016/j.tins.2004.05.012. URL http://dx.doi.org/10.1016/j.tins.2004.05.012. [p. 159]

Hippocrates. On the sacred disease, 400BC. [p. 1]

Hollup, S. A, Molden, S, Donnett, J. G, Moser, M. B, and Moser, E. I. Accumulation of hippocampal place fields at the goal location in an annular watermaze task. *The Journal of Neuroscience*, 21(5):1635–1644, 2001. [p. 161]

Holmes, P. The essence of chaos (en lorenz). *SIAM Review*, 37(1):129, 1995. [p. 48]

Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences USA*, 79:2554–2558, April 1982. [p. 3, 15, 34, 38, 60, 66, 81, 83, 90, 137, 181]

Hubel, D. H and Wiesel, T. N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology*, 160:106–154, 1962. [p. 15]

Huelse, M and Pasemann, F. Dynamical neural schmitt trigger for robot control. In Dorronsoro, J. R, editor, *ICANN 2002*, Springer Verlag Berlin Heidelberg New York, pages 783–788, 2002. [p. 62, 64]

Huerta, P. T and Lisman, J. E. Heightened synaptic plasticity of hippocampal ca1 neurons during a cholinergically induced rhythmic state.

Nature, 364(6439):723–725, Aug 1993. doi: 10.1038/364723a0. URL `http://dx.doi.org/10.1038/364723a0`. [p. 31]

I

Ierusalimschy, R, de Figueiredo, L. H, and Celes, W. *Lua 5.1 Reference Manual*. Lua.org, 2006. URL `http://www.lua.org/manual/5.1/`. [p. 78]

Ikeda, K, Matsumoto, K, and Otsuka, K. Maxwell-bloch turbulence. *Progress of Theoretical Physics Supplement*, 99:295–324, 1989. [p. 120]

J

Jackson, J. C, Johnson, A, and Redish, A. D. Hippocampal sharp waves and reactivation during awake states depend on repeated sequential experience. *The Journal of Neuroscience*, 26:12415–12426, 2006. [p. 32]

Jonides, J, Lewis, R. L, Nee, D. E, Lustig, C. A, Berman, M. G, and Moore, K. S. The mind and brain of short-term memory. *Annu Rev Psychol*, 59:193–224, 2008. doi: 10.1146/annurev.psych.59.103006.093615. URL `http://dx.doi.org/10.1146/annurev.psych.59.103006.093615`. [p. 22]

K

Kahana, M. J, Seelig, D, and Madsen, J. R. Theta returns. *Curr Opin Neurobiol*, 11(6):739–744, Dec 2001. [p. 31]

Kalinovsky, A and Scheiffele, P. Transcriptional control of synaptic differentiation by retrograde signals. *Curr Opin Neurobiol*, 14 (3):272–279, Jun 2004. doi: 10.1016/j.conb.2004.05.011. URL `http://dx.doi.org/10.1016/j.conb.2004.05.011`. [p. 159]

Kandel, E, Schwartz, J, and Jessel, T. *Principles of Neural Science*. McGraw-Hill Professional., 2000. [p. 10]

Kane, M. J and Engle, R. W. The role of prefrontal cortex in working-memory capacity, executive attention, and general fluid intelligence: an individual-differences perspective. *Psychon Bull Rev*, 9(4):637–671, Dec 2002. [p. 24]

Kaneko, K. Pattern dynamics in spatiotemporal chaos. *Physica D*, 34:1–41, 1992. [p. 120]

Kaneko, K and Tsuda, I. Chaotic itinerancy. *Chaos: Focus Issue on Chaotic Itinerancy*, 13(3):926–936, 2003. [p. 4, 120]

Kelso, J. A. S, Case, P, Holroyd, T, Horvath, E, Razaszek, J, Tuller, B, and Ding, M. *Multistability and metastability in perceptual and brain dynamics, Ambiguity in Mind and Nature : multistable cognitive phenomena*, volume 64. Springer Series in Synergetic, 1995. [p. 182]

Kenet, T, Bibitchkov, D, Tsodyks, M, Grinvald, A, and Arieli, A. Spontaneously emerging cortical representations of visual attributes. *Nature*, 425:954–956, october 2003. [p. 4, 81, 136]

Kentridge, R. W. Computation, chaos and non-deterministic symbolic computation: The chinese room problem solved ? *Psycoloquy - Symbolism Connectionism*, 12:17, 2000. [p. 84]

Kentros, C, Hargreaves, E, Hawkins, R. D, Kandel, E. R, Shapiro, M, and Muller, R. V. Abolition of long-term stability of new hippocampal place cell maps by nmda receptor blockade. *Science*, 280(5372):2121–2126, Jun 1998. [p. 161]

Kentros, C. G, Agnihotri, N. T, Streater, S, Hawkins, R. D, and Kandel, E. R. Increased attention to spatial context increases both place field stability and spatial memory. *Neuron*, 42(2):283–295, Apr 2004. [p. 161]

Kintsch, W, Patel, V, and Ericsson, A. The role of long-term working memory in text comprehension. *Psychologia*, 42:186–198, 1999. [p. 26]

Kirkpatrick, S, Gelatt, C. D, and Vecchi, M. P. Optimization by simulated annealing. *Science*, 220:671–680, 1983. [p. 35]

Kirkwood, A and Bear, M. F. Homosynaptic long-term depression in the visual cortex. *Journal of Neuroscience*, 14:3404–3412, 1994. [p. 15]

Kitajo, K, Nozaki, D, Ward, L. M, and Yamamoto, Y. Behavioral stochastic resonance within the human brain. *Physical Review Letters*, 90:1–4, 2003. [p. 5]

Kohonen, T. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982. [p. 6, 35]

Kohonen, T. The self-organizing map. *Neurocomputing*, 21:1–6, 1998. [p. 65, 93]

Kohonen, T. *Self-Organizing Maps*. Springer., third, extended edition edition, 2001. [p. 6]

Kolb, B and Whishaw, I. Q. *Fundamentals of human neuropsychology*. Worth Pub, 2008. [p. 11]

Kötter, R and Stephan, K. E. Useless or helpful? the "limbic system" concept. *Rev Neurosci*, 8(2):139–145, 1997. [p. 29]

Kozma, R. On the constructive role of noise in stabilizing itinerant trajectories. *Chaos, Special Issue on Chaotic Itinerancy*, 13(3):1078–1090, 2003. [p. 5]

Kozma, R and Freeman, W. J. Chaotic resonance - methods and applications for robust classification of noisy and variable pattern. *International Journal of Bifurcation and Chaos*, 11(6):1607–1629, 2001. [p. 63]

Kremer, K. *A field guide to Dynamical Recurrent Networks*. IEEE press, 2001. [p. 66]

L

Lamb, S. M. *Pathways of the Brain: The Neurocognitive Basis of Language*. John Benjamins Publishing Co, 1998. [p. 35]

Lashley, K. In search of the engram. *Society of Experimental Biology Symposium*, 4:454–482, 1950. [p. 33]

Levy, W. B and Steward, O. Temporal contiguity requirements for long term associative potentiation/depression in the hippocampus. *Neuroscience*, 8:791–797, 1983. [p. 5, 92]

Lewandowsky, S, Duncan, M, and Brown, G. D. A. Time does not cause forgetting in short-term serial recall. *Psychon Bull Rev*, 11(5):771–790, Oct 2004. [p. 22]

Lom, B, Cogen, J, Sanchez, A. L, Vu, T, and Cohen-Cory, S. Local and target-derived brain-derived neurotrophic factor exert opposing effects on the dendritic arborization of retinal ganglion cells in vivo. *J Neurosci*, 22(17):7639–7649, Sep 2002. [p. 159]

Lorente De N, R. Studies on the structure of the cerebral cortex. continuation of the study of the ammonic system. *J. Psychol. Neurol.*, 46:113–177, 1934. [p. 30]

Lorenz, E. N. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20(2):130–148, 1963. [p. 43]

Lubenov, E. V and Siapas, A. G. Hippocampal theta oscillations are travelling waves. *Nature*, 459(7246):534–539, May 2009. [p. 31]

Luck, S. J and Vogel, E. K. The capacity of visual working memory for features and conjunctions. *Nature*, 390(6657):279–281, Nov 1997. doi: 10.1038/36846. URL `http://dx.doi.org/10.1038/36846`. [p. 20]

M

Maass, W and Bishop, C. M. *Pulsed Neural Networks*. MIT-Press, Cambridge, 1998. [p. 37]

MacLean, J. N, Watson, B. O, Aaron, G. B, and Yuste, R. Internal dynamics determine the cortical response to thalamic stimulation. *Neuron*, 48(5):811–823, Dec 2005. doi: 10.1016/j.neuron.2005.09.035. URL `http://dx.doi.org/10.1016/j.neuron.2005.09.035`. [p. 136]

Maguire, E. A, Burgess, N, Donnett, J. G, Frackowiak, R. S, Frith, C. D, and O'Keefe, J. Knowing where and getting there: a human navigation network. *Science*, 280(5365):921–924, May 1998. [p. 29]

Maguire, E. A, Gadian, D. G.and Johnsrude, I. S, Good, C. D, Ashburner, J, Frackowiak, R. S, and Frith, C. D. Navigation-related structural change in the hippocampi of taxi drivers. *PNAS*, 97:4398–4403, 2000. [p. 29]

Malenka, R. C and Bear, M. F. Ltp and ltd: an embarrassment of riches. *Neuron*, 44:5–21, 2004. [p. 14]

Mandelbrot, B. *Les objets fractals*. Flammarion, Paris, 1975. [p. 43]

Mandler, G. *The psychology of learning and motivation: : Advances in research and theory*, volume 1, chapter Organization and memory, page 381. Oxford, England: Academic Press., 1967. [p. 23]

Mason, S. F. *A History of the Sciences*. MacMillan Publishing Company, 1962. [p. 1]

Matsumura, N, Nishijo, H, Tamura, R, Eifuku, S, Endo, S, and Ono, T. Spatial- and task-dependent neuronal responses during real and virtual translocation in the monkey hippocampal formation. *J Neurosci*, 19(6):2381–2393, Mar 1999. [p. 28]

McCulloch, W. S and Pitts, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943. [p. 33, 36, 69, 70]

McNaughton, B. L, Battaglia, F. P, Jensen, O, Moser, E. I, and Moser, M.-B. Path-integration and the neural basis of the cognitive map. *Nature Reviews Neuroscience*, 7:663–678, 2006. [p. 7]

Miller, E. K, Erickson, C. A, and Desimone, R. Neural mechanisms of visual working memory in prefrontal cortex of the macaque. *J Neurosci*, 16(16):5154–5167, Aug 1996. [p. 136]

Miller, G. A. The magical number seven plus or minus two: some limits on our capacity for processing information. *Psychol Rev*, 63(2):81–97, Mar 1956. [p. 20]

Miller, G. A, Galanter, E, and Pribram, K. H. *Plans and the Structure of Behavior*. Holt Rinehart and Winston, Inc., 1960. [p. 24]

Minsky, M and Papert, S. A. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, 1969. [p. 34]

Molter, C. *Storing information through complex dynamics in Recurrent Neural Networks*. PhD thesis, Université Libre de Bruxelles, 2005. [p. 112]

Molter, C and Bersini, H. How chaos in small hopfield networks makes sense of the world. *Proceedings of the IJCNN conference, Portland*, 2003a. [p. 3, 5, 7, 62, 83, 87, 93, 96, 131, 181, 182]

Molter, C and Bersini, H. Fascinating rhythms by chaotic hopfield networks. *Proceedings of the ECAL conference, Dortmund*, 2003b. [p. 3, 5, 7, 62, 83, 87, 93, 96, 131, 181, 182]

Molter, C, Salihoglu, U, and Bersini, H. How chaos boosts the encoding capacity of small recurrent neural networks: learning consideration. *Proceedings of the International Joint Conference on Neural Networks -IJCNN conference, Budapest*, 2004a. [p. 5, 8, 85, 97]

Molter, C, Salihoglu, U, and Bersini, H. Hetero-associative symbolic learning using neuromodules. Technical Paper, 2004b. [p. 8]

Molter, C, Salihoglu, U, and Bersini, H. Learning cycles brings chaos in continuous hopfield networks. *Proceedings of the International Joint Conference on Neural*

Networks -IJCNN conference, Montreal, 2005a. [p. 8]

Molter, C, Salihoglu, U, and Bersini, H. Introduction of an hebbian unsupervised learning algorithm to boost the encoding capacity of hopfield networks. *Proceedings of the International Joint Conference on Neural Networks -IJCNN conference, Montreal*, 2005b. [p. 8]

Molter, C, Salihoglu, U, and Bersini, H. Phase synchronization and chaotic dynamics in hebbian learned artificial recurrent neural networks. *CNS Workshop: Nonlinear spatio-temporal neural dynamics - Experiments and Theoretical Models*, 2005c. [p. 8]

Molter, C, Salihoglu, U, and Bersini, H. An interpretative recurrent neural network to improve pattern storing capabilities - dynamical considerations. *Proceedings of the International Symposium on Nonlinear Theory and Applications (NOLTA 2005), Brugge*, 2005d. [p. 8]

Molter, C, Salihoglu, U, and Bersini, H. Storing static and cyclic patterns in an hopfield neural network. Technical Paper, Jan 2005e. [p. 8]

Molter, C, Salihoglu, U, and Bersini, H. How to prevent spurious data in a chaotic brain. In Press, I, editor, *International Joint Conference on Neural Networks (IJCNN 2006)/. Proc WCCI*, pages 1365–1371, 2006a. [p. 6, 8, 124]

Molter, C, Sato, N, Salihoglu, U, and Yamaguchi, Y. How reward can induce reverse replay of behavioral sequences in the hippocampus. *Proceedings of ICONIP*, 2006b. [p. 8]

Molter, C, Salihoglu, U, and Bersini, H. How stochastic noise helps memory retrieval in a chaotic brain. In Press, I, editor, *International Joint Conference on Neural Networks (IJCNN 2007)*, pages 1365–1371, 2007a. [p. 7, 133]

Molter, C, Salihoglu, U, and Bersini, H. The road to chaos by time-asymmetric hebbian learning in recurrent neural networks. *Neural Computation*, 19(1):80–110, Jan 2007b. doi: 10.1162/neco.2007.19.1.80. URL `http://dx.doi.org/10.1162/neco.2007.19.1.80`. [p. 5, 7, 88, 97, 133]

Molter, C, Salihoglu, U, and Bersini, H. *Neurodynamics of Cognition and Consciousness*, chapter Giving meaning to cycles to go beyond the limitations of fixed point attractors, pages 305–324. Springer-Verlag Berlin and Heidelberg GmbH & Co. K, 2007c. [p. 7]

Molter, C, Colliaux, D, and Yamaguchi, Y. Working memory and spontaneous activity of cell assemblies. a biologically motivated computational model. *Proceedings of the International Joint Conference on Neural Networks - Hong Kong*, 2008. [p. 138, 139, 141]

Molter, C, Colliaux, D, and Yamaguchi, Y. Working memory dynamics and spontaneous activity in a flip-flop oscillations network model with milnor attractor. *Cognitive Neurodynamics*, 2:(in press), 2009. [p. 6, 137, 184]

Mongillo, G, Barak, O, and Tsodyks, M. Synaptic theory of working memory. *Science*, 319(5869):1543–1546, Mar 2008. doi: 10.1126/science.1150769. URL `http://dx.doi.org/10.1126/science.1150769`. [p. 6, 137, 138, 140, 141, 184]

Morris, R. G, Garrud, P, Rawlins, J. N, and O'Keefe, J. Place navigation impaired in rats with hippocampal lesions. *Nature*, 297(5868):681–683, Jun 1982. [p. 29]

Moser, E. I, Kropff, E, and Moser, M.-B. Place cells, grid cells, and the brain's spatial representation system. *Annu Rev Neurosci*, 31:69–89, 2008. doi: 10.1146/annurev.neuro.31.061307.090723. URL `http://dx.doi.org/10.1146/annurev.neuro.31.061307.090723`. [p. 28, 29]

Moser, M. B and Moser, E. I. Functional differentiation in the hippocampus. *Hippocampus*, 8(6):608–619, 1998. doi: gt;3.0.CO;2-7. URL `http://dx.doi.org/gt;3.0.CO;2-7`. [p. 29]

Moss, F, Ward, L. M, and Sannita, W. G. Stochastic resonance and sensory information processing: a tutorial and review of application. *Clinical Neurophysiology*, 115(2):267–281, 2004. [p. 5]

Muthu, S, Kozma, R, and Freeman, W. J. Applying kiv dynamic neural network model for real time navigation by mobile robot aibo. In *IEEE/INNS 2004 Int. Joint Conference on Neural Networks IJCNN04*, pages 1617–1622, Budapest, Hungary, July 2004. IEEE Press. [p. 63]

N

Nadel, L, O'Keefe, J, and Black, A. Slam on the brakes: a critique of altman, brunner, and bayer's response-inhibition model of hippocampal function. *Behav Biol*, 14(2):151–162, Jun 1975. [p. 27]

Nairne, J. S. Remembering over the short-term: the case against the standard model. *Annu Rev Psychol*, 53:53–81, 2002. doi: 10.1146/annurev.psych.53.100901.135131. URL `http://dx.doi.org/10.1146/annurev.psych.53.100901.135131`. [p. 22]

Nairne, J. S and Dutta, A. Spatial and temporal uncertainty in long-term memory. *Journal of Memory and Language*, 31:396–407, 1992. [p. 21]

Neuenschwander, S, Engel, A, Konig, P, Singer, W, and Varela, F. Synchronization of neuronal responses in the optic tectum of awake pigeons. *Vis. Neurosci.*, 13: 575–584, 1996. [p. 17]

Nickel, M, Vitello, M, and Brümmer, F. Dynamics and cellular movements in the locomotion of the sponge tethya wilhelma. *Integr Comp Biol*, 42:1285, 2002. [p. 9]

Nicolis, J and Tsuda, I. Chaotic dynamics of information processing. the 'magic number seven plusminus two' revisited. *Bulletin of Mathematical Biology*, 47: 343–365, 1985. [p. 3, 18, 81]

Nunez, P. L. *Electric Fields of the Brain*. Oxford University Press, New York, 1981. [p. 17]

O

Oberauer, K. Access to information in working memory: exploring the focus of attention. *J Exp Psychol Learn Mem Cogn*, 28(3):411–421, May 2002. [p. 25]

Oberauer, K and Kliegl, R. A formal model of capacity limits in working memory. *Journal of Memory and Language*, 55:601–626, 2006. [p. 22]

O'Keefe, J and Dostrovsky, J. The hippocampus as a spatial map. preliminary evidence from unit activity in the freely-moving rat. *Brain Research*, 34(1):171–175, Nov 1971. [p. 1, 28]

O'Keefe, J and Nadel, L. *The Hippocampus as a Cognitive Map*. Oxford University Press, 1978. [p. 28, 29]

Omlin, C. Understanding and explaining drn behavior. In J. Kolen, S. K, editor, *A field guide to Dynamical Recurrent Networks*, chapter 12. IEEE press, 2001. [p. 67, 83]

Oppenheim, R. W. Cell death during development of the nervous system. *Annu Rev Neurosci*, 14:453–501, 1991. doi: 10.1146/annurev.ne.14.030191.002321. URL `http://dx.doi.org/10.1146/annurev.ne.14.030191.002321`. [p. 6, 159]

Oseledets, V. I. A multiplicative ergodic theorem. lyapunov characteristic numbers for dynamical systems. *Trans.Moscow Math. Soc.*, 19:197–231, 1968. [p. 51]

Ott, E. *Chaos in Dynamical Systems*. Cambridge University Press, 1993. [p. 41]

P

Palmere, M, Benton, S. L, Glover, J. A, and Ronning, R. R. Elaboration and recall of main ideas in prose. *Journal of Educational Psychology.*, 75(6):898–907, 1983. [p. 24]

Parmentier, F. B. R, Elford, G, and Mayberry, M. Transitional information in spatial serial memory: path characteristics affect recall performance. *J Exp Psychol Learn Mem Cogn*, 31(3):412–427, May 2005. doi: 10.1037/0278-7393.31.3.412. URL http://dx.doi.org/10.1037/0278-7393.31.3.412. [p. 20]

Pasemann, F. Dynamics of a single model neuron. *International Journal of Bifurcations and Chaos*, 2:271–278, 1993. [p. 59, 64]

Pasemann, F. A simple chaotic neuron. *Physica D*, 104:205–211, 1997. [p. 60, 62, 64]

Pasemann, F. Synchronized chaos and other coherent states for two coupled neurons. *Physica D*, 128:236–249, 1999. [p. 60, 64]

Pasemann, F. Complex dynamics and the structure of small neural networks. *Network: Computation in neural systems*, 13(2):195–216, 2002. [p. 4, 64]

Pasemann, F, Hild, M, and Zahedi, K. So(2)-networks as neural oscillators. In J.Mira and R.Alvarez, J, editors, *Computational Methods in Neural Modeling*, volume LNCS 2686 of *Proceedings IWANN 2003*, pages 144–151. Springer, 2003. [p. 60, 64]

Pastalkova, E, Itskov, V, Amarasingham, A, and Buzsáki, G. Internally generated cell assembly sequences in the rat hippocampus. *Science*, 321(5894):1322–1327, Sep 2008. doi: 10.1126/science.1159775. URL http://dx.doi.org/10.1126/science.1159775. [p. 138]

Paus, T, Collins, D. L, Evans, A. C, Leonard, G, Pike, B, and Zijdenbos, A. Maturation of white matter in the human brain: a review of magnetic resonance studies. *Brain Res Bull*, 54(3):255–266, Feb 2001. [p. 12]

Pelvig, D. P, Pakkenberg, H, Stark, A. K, and Pakkenberg, B. Neocortical glial cell numbers in human brains. *Neurobiol Aging*, 29(11): 1754–1762, Nov 2008. doi: 10.1016/j.neurobiolaging.2007.04.013. URL http://dx.doi.org/10.1016/j.neurobiolaging.2007.04.013. [p. 9]

Peterson, L. R and Peterson, M. J. Short-term retention of individual verbal items. *J Exp Psychol*, 58:193–198, Sep 1959. [p. 22]

Piaget, J. *The Psychology of Intelligence*. Routledge, New York, 1963. [p. 93]

Poirier, M and Saint-Aubin, J. Memory for related and unrelated words: further evidence on the influence of semantic factors in immediate serial recall. *Q J Exp Psychol A*, 48(2):384–404, May 1995. [p. 20]

Poirier, M and Saint-Aubin, J. Immediate serial recall, word frequency, item identity and item position. *Can J Exp Psychol*, 50(4):408–412, Dec 1996. [p. 20]

Pollack, J. B. Connectionism: Past, present, and future. *Artificial Intelligence Review*, 1988. [p. 34]

Pomeau, Y and Manneville, P. Intermittent transitions to turbulence in dissipative dynamical systems. *Comm. Math. Phys.*, 74:189–197, 1980. [p. 122]

Purves, D and Lichtman, J. W. *Principles of neural development*. Sinauer Associates, 1985. [p. 12]

R

Rainer, G, Asaad, W. F, and Miller, E. K. Selective representation of relevant information by neurons in the primate prefrontal cortex. *Nature*, 393:577–579, 1998. [p. 6, 136]

Ramón y Cajal, S. The croonian lecture: La fine structure des centres nerveux. *Proceedings of the Royal Society of London*, 55:444–468, January 1894. [p. 14]

Rapp, P. Chaos in the neurosciences: Cautionary tales from the frontier. *Biologist*, 40:89–94, 1993. [p. 18]

Ridley, M. *The agile gene: how nature turns on nurture*. Perennial, 2004. [p. 12]

Robertson, E. M, Pascual-Leone, A, and Press, D. Z. Awareness modifies the skill-learning benefits of sleep. *Curr Biol*, 14(3):208–212, Feb 2004. doi: 10.1016/j.cub.2004.01.027. URL `http://dx.doi.org/10.1016/j.cub.2004.01.027`. [p. 23]

Rodriguez, E, George, N, Lachaux, J. P, Renault, B, Martinerie, J, Reunault, B, and Varela, F. J. Perception's shadow: long-distance synchronization of human brain activity. *Nature*, 397:430–433, 1999. [p. 3, 17, 81, 183]

Rolls, E. T and Treves, A. *Neural Networks and Brain Function*. Oxford University Press, Oxford, 1998. [p. 77]

Rolls, E. T and Xiang, J. Z. Spatial view cells in the primate hippocampus and memory recall. *Rev Neurosci*, 17(1–2):175–200, 2006. [p. 28]

Rosenblatt, F. *Principles of Neurodynamics, Perceptrons and the Theory of Brain Mechanisms*. Spartan, New York, 1962. [p. 15, 34]

Rössler, O. E. The chaotic hierarchy. *Zeitschrift fr Naturforschung*, 38a:788–801, 1983. [p. 51, 120]

Ruelle, D and Takens, F. On the nature of turbulence. *Commun. Math. Phys.*, 20:167–192, 1971. [p. 43, 55]

Rumelhart, D. E, McClelland, J. L, and the PDP Research Group. *Parallel Distributed Processing : Foundations*, volume 1. MIT Press, 1986a. [p. 15, 34, 35, 85, 185]

Rumelhart, D. E, McClelland, J. L, and the PDP Research Group. *Parallel Distributed Processing : Explorations in the Microstructure of Cognition*. MIT Press, 1986b. [p. 15, 34, 35, 85, 185]

Russel, D. F, Wilkens, L. A, and Moss, F. Use of behavioural stochastic resonance by paddle fish for feeding. *Nature*, 402:291–294, 1999. [p. 5]

S

Salihoglu, U. Chaos in small recurrent neural networks: theoretical and practical studies. Master's thesis, Université Libre de Bruxelles, 2004. [p. 8]

Salihoglu, U, Molter, C, and Bersini, H. How stochastic noise helps memory retrieval in a chaotic brain. In *Proc. International Joint Conference on Neural Networks IJCNN 2007*, pages 1458–1463, 12–17 Aug. 2007. doi: 10.1109/IJCNN.2007.4371173. [p. 5, 128, 133]

Salihoglu, U, Molter, C, and Bersini, H. Visual scene identification through complex dynamics in recurrent neural networks to solve the binding problem. *Proceedings of the ISNN conference*, 2008. [p. 7]

Salihoglu, U, Bersini, H, Yamaguchi, Y, and Molter, C. A model for the cognitive map formation : Application of the retroaxonal theory. *Proc. IEEE International Joint Conference on Neural Networks*, 2009a. [p. 7]

Salihoglu, U, Bersini, H, Yamaguchi, Y, and Molter, C. Online unsupervised formation of cell assemblies for the encoding of multiple cognitive maps. *Neural Networks*, 22:687–696, 2009b. [p. 7, 149, 173, 180]

Samsonovich, A and McNaughton, B. L. Path integration and cognitive mapping in a continuous attractor neural network model. *The Journal of Neuroscience*, 17(15):5900–5920, 1997. [p. 7]

Scoville, W. B and Milner, B. Loss of recent memory after bilateral hippocampal lesions. *Journal of Neurology, Neurosurgery and Psychiatry*, 20:11–21, 1957. [p. 27]

Searle, J. *Minds, Brains, and Science*. Harvard University Press, Cambridge, 1984. [p. 2]

Searle, J. *The Rediscovery of the Mind*. MIT Press, Cambridge, 1992. [p. 3]

Sejnowski, T and Rosenberg, C. Parallel networks that learn to pronounce english text. *Complex Systems*, 1:145–168, 1987. [p. 15]

Sejnowski, T. J. Storing covariance with nonlinearly interacting neurons. *J. Math. Biol.*, 4:303, 1977. [p. 81]

Silberberg, G, Bethge, M, Markram, H, Pawelzik, K, and Tsodyks, M. Dynamics of population rate codes in ensembles of neocortical neurons. *J Neurophysiol*, 91: 704–709, 2004. [p. 5]

Simon, H. A. How big is a chunk?: By combining data from several experiments, a basic human memory unit can be identified and measured. *Science*, 183(4124):482–488, Feb 1974. doi: 10.1126/science.183.4124.482. URL http://dx.doi.org/10.1126/science.183.4124.482. [p. 20]

Sinha, S and Ditto, W. L. Computing with distributed chaos. *Physical Review E*, 60:363–377, 1999. [p. 183]

Skaggs, W. E, McNaughton, B. L, Permenter, M, Archibeque, M, Vogt, J, Amaral, D. G, and Barnes, C. A. Eeg sharp waves and sparse ensemble unit activity in the macaque hippocampus. *J Neurophysiol*, 98(2):898–910, Aug 2007. doi: 10.1152/jn.00401.2007. URL http://dx.doi.org/10.1152/jn.00401.2007. [p. 33]

Skarda, C. A and Freeman, W. J. How brains make chaos in order to make sense of the world. *Behavioral and Brain Sciences*, 10:161–195, 1987. [p. 3, 4, 17, 18, 63, 81, 84, 181]

Skarda, C. A and Freeman, W. J. *John Searle and his Critics*, chapter Mind/Brain Science: Neuroscience on philosophy of mind, pages 115–127. Blackwell, Oxford UK, 1990a. [p. 16, 17, 18]

Skarda, C. A and Freeman, W. J. Chaos and the new science of the brain. In *Concepts in Neuroscience*, volume 1, pages 275–285. World Scientific Publishing Company, 1990b. [p. 17, 18, 65, 84, 183]

Smith, D. M and Mizumori, S. J. Y. Learning-related development of context-specific neuronal responses to places and events: the hippocampal role in context processing. *J Neurosci*, 26(12):3154–3163, Mar 2006. doi: 10.1523/JNEUROSCI.3234-05.2006. URL http://dx.doi.org/10.1523/JNEUROSCI.3234-05.2006. [p. 29]

Solstad, T, Boccara, C. N, Kropff, E, Moser, M.-B, and Moser, E. I. Representation of geometric borders in the entorhinal cortex. *Science*, 322(5909):1865–1868, Dec 2008. doi: 10.1126/science.1166466. URL http://dx.doi.org/10.1126/science.1166466. [p. 29]

Sompolinsky, H, Crisanti, A, and Sommers, H. J. Chaos in random neural networks. *Physical Review Letters*, 61:258–262, 1988. [p. 18]

Spencer, H. *The principles of psychology*. Appleton, 1872. [p. 35]

Sperling, G. A. The information available in brief visual persentation. *Psychological Monographs : General and Applied*, 74:1–29, 1960. [p. 19]

Squire, L. R. Memory and the hippocampus: A synthesis from findings with rats, monkeys, and humans. *Psychological Review*, 99:195–231, 1992. [p. 28]

Squire, L. R. The legacy of patient h. m. for neuroscience. *Neuron*, 61(1):6–9, Jan 2009. doi: 10.1016/j.neuron.2008.12.023. URL http://dx.doi.org/10.1016/j.neuron.2008.12.023. [p. 21, 27]

Squire, L. R and Schacter, D. L. *The Neuropsychology of Memory*. Guilford Press, 2002. [p. 28]

Sun, R and Alexandre, F. *Connectionist-Symbolic Integration: From Unified to Hybrid Approaches*. Lawrence Erlbaum, 1997. [p. 35]

Sutherland, G. R and McNaughton, B. Memory trace reactivation in hippocampal and neocortical neuronal ensembles. *Curr Opin Neurobiol*, 10(2):180–186, Apr 2000. [p. 33]

T

Tao, H, Zhang, L. I, Bi, G, and Poo, M. Selective presynaptic propagation of long-term potentiation in defined neural networks. *J Neurosci*, 20(9):3233–3243, May 2000. [p. 159]

Tarnow, E. Short term memory may be the depletion of the readily releasable pool of presynaptic neurotransmitter vesicles. *Cogn Neurodyn*, May 2008a. doi: 10.1007/s11571-009-9085-1. URL `http://dx.doi.org/10.1007/s11571-009-9085-1`. [p. 21]

Tarnow, E. Response probability and latency: a straight line, an operational definition of meaning and the structure of short term memory. *Cogn Neurodyn*, 2008b. [p. 21]

Thomasson, N, Pezard, L, F., A. J, Renault, B, and Martinerie, J. Nonlinear eeg changes associated with clinical improvement in depressed patients. *Nonlinear Dynamics in Psychology and Life Sciences*, 4(3):203–218, 2000. [p. 18]

Tolman, E. C. Cognitive maps in rats and men. *Psychol Rev*, 55(4):189–208, Jul 1948. [p. 28]

Tsuda, I. Dynamic link of memorychaotic memory map in nonequilibrium neural networks. *Neural Networks*, 5:313–326, 1992. [p. 120]

Tsuda, I. Towards an interpretation of dynamic neural activity in terms of chaotic dynamical systems. *Behavioral and Brain Sciences*, 24, 2001. [p. 18, 84, 93, 136, 183]

Tulving, E. Episodic memory: from mind to brain. *Annual review of psychology*, 53:1–25, 2002. [p. 25]

Tyler, S. W, Hertel, P. T, McCallum, M. C, and Ellis, H. C. Cognitive effort and memory. *Journal of Experimental Psychology: Human Learning and Memory*, 5 (6):607–617, 1979. [p. 24]

U

Usher, M and Feingold, M. Stochastic resonance in the speed of memory retrieval. *Biological Cybernetics*, 83(6):L011–L016, 2000. [p. 184]

Uwate, Y and Nishio, Y. Back propagation learning of neural networks with chaotically-selected affordable neurons. *IEEE International Symposium on Circuits and Systems*, 2:1481–1484, 2005. [p. 5]

V

van Hemmen, J and Kuhn, R. *Models of Neural Networks*, chapter Storage Capacity, pages 1–113. Springer, 2nd edition, 1995. [p. 90]

van Hemmen, J. L and Sejnowski, T. J. *23 Problems in Systems Neuroscience*. Oxford University Press, 2005. [p. 10]

van Praag, H, Kempermann, G, and Gage, F. H. Neural consequences of environmental enrichment. *Nat Rev Neurosci*, 1(3):191–198, Dec 2000. [p. 13]

van Vreeswijk, C and Sompolinsky, H. Chaos in neuronal networks with balanced excitatory and inhibitory activity. *Science*, 274(5293):1724–1726, 1996. [p. 5]

Vanderwolf, C. H. The hippocampus as an olfacto-motor mechanism: were the classical anatomists right after all? *Behav Brain Res*, 127(1–2):25–47, Dec 2001. [p. 27]

Vanderwolf, C. H. Hippocampal electrical activity and voluntary movement in the rat. *Electroencephalography and Clinical Neurophysiology*, 26:407–418, 1969. [p. 31]

Varela, F. Resonnant cell assemblies: a new approach to cognitive function and neuronal synchrony. *Biol.Res.*, 28:81–95, 1995. [p. 17]

Varela, F, Thompson, E, and Rosch, E. *The Embodied Mind: Cognitive Science and Human Experience*. MIT Press, 1991. [p. 93, 183]

Vincent, J. L and Buckner, R. L. Unrest at rest: default activity and spontaneous network correlations. *Neuroimage*, 37(4):1091–1096, 2007. [p. 136]

W

Wagner, U, Gais, S, Haider, H, Verleger, R, and Born, J. Sleep inspires insight. *Nature*, 427(6972):352–355, Jan 2004. doi: 10.1038/nature02223. URL `http://dx.doi.org/10.1038/nature02223`. [p. 23]

Werbos, P. *Beyond Regression New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, 1974. [p. 39, 85, 159, 185]

Werbos, P. J. *The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting*. Wiley-Interscience, 1994. [p. 39]

Widrow, B and Hoff, M. E. Adaptive switching circuits. *IRE WESCON Convention Record*, pages 96–104, 1960. [p. 37]

Wiesel, T. N. Postnatal development of the visual cortex and the influence of environment. *Nature*, 299(5884):583–591, Oct 1982. [p. 12]

Wiesenfeld, K and Moss, F. Stochastic resonance and the benefits of noise: from ice ages to crayfish and squids. *Nature*, 373:33–36, 1995. [p. 5]

Williams, R. J and Zipser, D. Gradient-based learning algorithms for recurrent connectionnist networks. *NU-CCS-90-9*, 1990. [p. 85]

Wilson, M. A and McNaughton, B. L. Reactivation of hippocampal ensemble memories during sleep. *Science*, 265(5172):676–679, Jul 1994. [p. 32]

Winson, J. Loss of hippocampal theta rhythm results in spatial memory deficit in the rat. *Science*, 201(4351):160–163, Jul 1978. [p. 31]

Wolf, A, Swift, J. B, Swinney, H, and Vastano, J. A. Determining lyapunov exponents from a time series. *Physica*, D16(D16):285–317, 1984. [p. 50, 52, 112]