Storing information through complex dynamics in Recurrent Neural Networks

— Tchouang-Tseu - PHILOSOPHES TAOÏSTES - BIBLIOTHEQUE DE LE PLEI-ADE PP.142
Abstract

The neural net computer simulations which will be presented here are based on the acceptance of a set of assumptions that for the last twenty years have been expressed in the fields of information processing, neurophysiology and cognitive sciences. First of all, neural networks and their dynamical behaviors in terms of attractors is the natural way adopted by the brain to encode information. Any information item to be stored in the neural net should be coded in some way or another in one of the dynamical attractors of the brain and retrieved by stimulating the net so as to trap its dynamics in the desired item’s basin of attraction. The second view shared by neural net researchers is to base the learning of the synaptic matrix on a local Hebbian mechanism. The last assumption is the presence of chaos and the benefit gained by its presence. Chaos, although very simply produced, inherently possesses an infinite amount of cyclic regimes that can be exploited for coding information. Moreover, the network 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 coming stimulus.

In this thesis, it is shown experimentally that the more information is to be stored in robust cyclic attractors, the more chaos appears as a regime in the back, erratically itinerating among brief appearances of these attractors. Chaos does not appear to be the cause but the consequence of the learning. However, it appears as an helpful consequence that widens the net’s encoding capacity. To learn the information to be stored, an unsupervised Hebbian learning algorithm is introduced. By leaving the semantics of the attractors to be associated with the feeding data unprescribed, promising results have been obtained in term of storing capacity.
Remerciements

Je suis tout d’abord particulièrement reconnaissant aux rapporteurs et à chacun des membres du jury pour l’honneur qu’ils m’ont fait en acceptant d’évaluer mon travail. Ainsi mes remerciements vont à Philippe Van Ham, Nicolas Cerf, Pasquale Nardone, Marceline Kaufman, Pierre Gaspard et enfin Robert Kozma que je remercie tout particulièrement pour avoir accepté de se déplacer dans notre pays lointain pour la défense de cette thèse.

Je suis tout particulièrement reconnaissant envers Hugues Bersini pour m’avoir accepté dans son service ainsi que pour avoir supervisé ma recherche. Il faut savoir que je me suis retrouvé propulsé dans ce monde bizarre de l’intelligence artificielle, monde à la confluence de diverses disciplines (de la science, des mathématiques, de la philosophie et d’autres disciplines plus esotériques encore), monde fourmillant de chercheurs en quête du graal de l’intelligence, de façon très inopinée. Hugues Bersini m’a initié à ce monde et à ses règles tout en me laissant la plus entière liberté dans mes choix scientifiques, chose pour laquelle je l’en remercie vivement. Il m’a cependant à plusieurs reprises contraint à effectuer des choix et à aller de l’avant, à travers des remarques, parfois très anodines, toujours très pertinentes (d’autant que je m’en souviens), tout en sachant, en adepte du chaos, qu’il est souvent illusoire de prétendre à pouvoir déterminer voire contrôler une trajectoire.

Je remercie très vivement également Utku Salihoglu pour avoir partagé ma recherche durant ces deux dernières années. Tandis que je me croyais libre développeur de mondes, par le biais de Java, il m’a ouvert les yeux en me faisant découvrir les richesses du C++ et de sa générativité. L’ imposture apparaître criante, Java n’était qu’un fils indigne au fabuleux royaume de l’orienté-objet. Plus sérieusement, cette rencontre a donné une autre dimension à ma recherche, à travers de nombreuses discussions, différentes nouvelles idées ont vu le jour, et une grande partie des résultats présentés dans cette thèse n’aurait jamais eu lieu sans sa contribution. Je remercie de même l’ensemble des personnes constituant le service iridia, et plus particulièrement, Marco Dorigo, l’autre personnalité importante de ce service bicéphale, Bruno Marchal, pour ses réflexions logiques qui mettent le monde à l’envers (et il en a souvent besoin), et Muriel Decreton, pour maintenir tout ce petit monde en place.

Il a été décidé que cette thèse serait écrite en anglais. Ce pour ne pas déroger à l’ordre des choses: il est communément admis que la communication scientifique doit être réalisée par ce biais, pour peu que l’on désire être entendu. Cependant ce choix n’allait pas sans difficultés, en témoignerait la mécompréhension de plusieurs collègues scientifiques lors de la lecture de mes précédentes publications.

Je suis dès lors très reconnaissant envers plusieurs personnes pour avoir
contribué à améliorer la clarté, la lisibilité et la fluidité de cette thèse. Ainsi je remercie Alan Coletta pour avoir contribué à rendre certaines expressions anglaises un peu moins “frenchy”. De même je remercie David Weiss pour ces pertinentes remarques attrapées au vol lors de son départ au ski. Écrire en anglais m’a également amené à plusieurs reprises à perdre tout fil méthodologique. Il est en effet déjà difficile dans sa propre langue maternelle de se relire de manière neutre, sans ajouter inconsciemment le sens manquant aux phrasés incomplets. Ce phénomène, j’ai pu le constater, s’amplifie dans les autres langues. Ainsi, je remercie Nathanael Ackerman pour ses nombreuses et avisées suggestions méthodologiques. Je remercie de même Tom Leenaerts pour sa contribution décisive à la clarté de certains chapitres.

Cette thèse m’a en outre permis de reprendre contact avec un ami exilé. Je remercie ainsi Stéphane Coen pour avoir aidé (et plus encore) au formatage de cette thèse.

Enfin je tiens à exprimer toute ma gratitude envers Juanita Maes pour avoir lu et relu l’entièreté de ce travail. Non seulement afin d’en extraire les nombreuses fautes d’orthographe, mais surtout et avant tout afin de la traduire “d’anglais à anglais”. Cette relecture unique a en outre permis de maintenir une cohérence globale. En ne dépendant pas uniquement de multiples relectures partielles, l’écueil classique d’une perte de sens a pu être évité. Un tel problème est bien connu en intelligence artificielle où, lors de ses premiers balbutiements, après multiples passes dans un programme de traduction automatique, la phrase “The spirit is willing, but the flesh is weak” était traduite par “The vodka is tempting, but the meat is a bit suspect”.

Je remercie enfin Coralie, mes parents et grand parents, mes sœurs, mes frères, mes nièces, ainsi que la plupart de mes amis. Et, fidèle à la théorie du Chaos, je remercie l’ensemble des personnes qui ont participés à ces événements fortuits de la vie qui m’ont conduit à cette présentation. Et puisque les mots ne touchent que les hommes, je m’arrêterai ici.

—— Colin Molter - BRUXELLES, MAI 2005
## Contents

1 **Introduction** .................................................. 3

2 **Preliminary Notions** ......................................... 9

   1 Introduction .................................................. 9

   2 Neurophysiology ............................................. 9

       2.1 Elementary notions of neuroanatomy ................... 10

       2.2 The nerve cell: the Neuron ............................ 12

3 Neurodynamics and the physiological basis of behavior ........ 16

       3.1 The neuron doctrine .................................. 16

       3.2 The connectionist approach .......................... 16

       3.3 The dynamical approach ............................... 17

       3.4 Chaos, Brain and Cognition .......................... 20

4 **Connectionist Models** ...................................... 22

   4.1 History .................................................. 23

   4.2 Structure of Connectionist Models ...................... 24

5 Dynamical systems theory, chaos and recurrent neural networks . 29

   5.1 History .................................................. 29

   5.2 Definition and general properties ....................... 31

   5.3 Theoretical application of the Dynamical System Theory to RNNs . 38

   5.4 Tools used ............................................. 43

   5.5 Initial conditions, basins of attractions and hysteresis .... 47

6 **Conclusion** .................................................. 49

3 **State of the art and Methodology** .......................... 51

   1 Introduction ............................................. 51

   2 Related approaches ..................................... 51

       2.1 Freeman and Kozma et al. ............................. 51

       2.2 Doyon and Samuelides et al. .......................... 52

       2.3 Pasemann et al. ..................................... 52

3 Our approach .................................................. 53

       3.1 What “information” means in this thesis ............... 54

       3.2 Description of the model ............................ 56

       3.3 The methodology used: development of a neural toolbox ... 58
# CONTENTS

## 4 Dynamical observations of random networks

1. Introduction ........................................... 61
2. Storing information in the dynamical attractors ................................. 61
   2.1 An infinite potential ....................................... 61
   2.2 Model Quantization ....................................... 63
   2.3 Example .............................................. 64
   2.4 Model’s Capacity ....................................... 64
3. Numerical results of encoding capacity .......................................... 66
   3.1 Quantitative Results ..................................... 66
   3.2 Qualitative Results ..................................... 75
4. Conclusion ............................................ 76

## 5 Learning using a gradient based algorithm

1. Introduction ........................................... 79
2. Description of the learning task ............................................... 80
3. Description of the BPTT algorithm ........................................ 80
   3.1 Introduction ........................................... 80
   3.2 The backpropagation algorithm ................................ 81
   3.3 Backpropagation Through Time Algorithm ......................... 84
   3.4 Learning stimulus: practical considerations ....................... 84
   3.5 Alternatives to the classical BPTT algorithm ...................... 85
4. Numerical Results ........................................ 86
   4.1 Efficiency of the BPTT ..................................... 87
   4.2 Dynamics of network after learning ................................ 88
   4.3 Weight distributions after learning ................................ 89
5. Conclusion ............................................ 91

## 6 Learning using an iterative supervised Hebbian algorithm

1. Introduction ........................................... 93
2. Description of the learning task and of the model .............................. 94
   2.1 The learning task ........................................ 95
   2.2 The model ............................................. 96
   2.3 Description of the algorithm ................................ 97
   2.4 The perceptron version of the Hebbian learning ...................... 97
   2.5 Learning cycles ....................................... 99
3. Experiments and Results .................................... 99
   3.1 Results on auto-associative memories .......................... 101
   3.2 Results on hetero-associative memories ......................... 102
   3.3 Learning Sequences of patterns ................................ 104
   3.4 Learning incrementally: relying on the iterative process .......... 105
4. Conclusion ............................................ 106
## 7 Dynamical observations on learned networks

1. Introduction ............................................. 109
2. Modifications entailed by the use of continuous neurons .......... 110
   2.1 Model’s modification: use of a sigmoid-like activation function .... 110
   2.2 Adaptation of the algorithm to continuous activation functions .... 110
3. Experiments and Results .................................. 112
   3.1 Weight distributions after learning .......................... 112
   3.2 Hebbian road to stability .................................. 113
   3.3 Hebbian learning of cycles: a new road to chaos .............. 116
   3.4 Characterization of the chaotic dynamics ...................... 119
4. Conclusion .............................................. 121

## 8 Learning using an iterative unsupervised Hebbian algorithm

1. Introduction ............................................. 123
2. The unsupervised Hebbian learning algorithm ....................... 124
   2.1 Description of the unsupervised learning task .................. 124
   2.2 The iterative supervised Hebbian algorithm ................... 125
   2.3 The iterative unsupervised Hebbian algorithm ................ 126
3. Experiments and Results .................................. 128
   3.1 Performance of this algorithm ................................ 128
   3.2 Weight distribution after learning ........................... 129
   3.3 Lyapunov exponent analysis and probability of chaos ........... 129
   3.4 Power spectrum and return map analysis ....................... 132
4. Conclusion .............................................. 133

## 9 The frustrated Chaos

1. Introduction ............................................. 135
2. Summary of the chaotic dynamics encountered ...................... 136
   2.1 Chaos in random RNNs .................................... 136
   2.2 Chaos in Hebbian learned RNNs ............................ 137
3. Description of the frustrated chaos ................................ 137
   3.1 Qualitative analyses ....................................... 137
   3.2 Quantitative analyses ....................................... 138
   3.3 Frustrated chaos related to chaotic itinerancy ................ 138
4. Conclusion .............................................. 140

## 10 Conclusion

1. Development of a Neural Development Kit .......................... 147
2. Introduction ............................................. 147
2.1 Runtime requirements ...................................... 148
2.2 Implementation requirements .................................. 148
3. Analyses .................................................. 149
   3.1 Analysis of the runtime requirements ........................ 149
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2 Analysis of the implementation requirements</td>
<td>150</td>
</tr>
<tr>
<td>4 Design</td>
<td>150</td>
</tr>
<tr>
<td>5 Practical aspects</td>
<td>151</td>
</tr>
<tr>
<td>Bibliography</td>
<td>156</td>
</tr>
</tbody>
</table>
List of Figures

2 Preliminary Notions
  2.1 Basic description of a neuron and its action potential .......................... 13
  2.2 EEG measures and tracing ................................................................. 17
  2.3 Comparison between a nerve cell and a formal neuron ......................... 25
  2.4 Comparison between a feedforward and a recurrent network ................. 26
  2.5 Illustration of the three major kinds of bifurcations ........................... 36
  2.6 Period doubling road to Chaos .......................................................... 39
  2.7 Quasi-periodic road to Chaos ............................................................ 40
  2.8 Intermittency Chaos ............................................................................ 41
  2.9 Stability domains for the one neuron network ...................................... 42
  2.10 Illustration of the hysteresis phenomenon for the one-neuron RNN ........ 48
  2.11 Illustration of the hysteresis phenomenon for the three-neuron RNN ....... 48

3 State of the art and Methodology
  3.1 Basins of attraction in the classical Hopfield model ............................... 55
  3.2 Illustration of the recurrent model used ............................................... 58

4 Dynamical observations of random networks
  4.1 Illustration of the influence of a stimulus to the mean signal ................. 62
  4.2 Illustration of the quantization of the model ........................................ 65
  4.3 General result: encoding capacity versus complexity in random networks 67
  4.4 Influence of the network’s size ............................................................ 68
  4.5 Influence of the network’s size on the mean signal ................................ 70
  4.6 Influence of the weight distribution ..................................................... 71
  4.7 Influence of the weight distribution ..................................................... 72
  4.8 Influence of the slope parameter $g$ of the activation function ............... 72
  4.9 Influence of the expected robustness .................................................. 73
  4.10 Weight distribution of a network having a huge potential ................... 74
  4.11 Weight distribution of a network having a small complexity ............... 74
  4.12 Chaotic dynamics in between robust limit cycle attractors ................. 77

5 Learning using a gradient based algorithm
  5.1 A very classical 1-layered perceptron .................................................. 81
<table>
<thead>
<tr>
<th>FIGURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2</td>
<td>Unfolding process of the model</td>
</tr>
<tr>
<td>5.3</td>
<td>Illustration of the importance of tuning the learning rate</td>
</tr>
<tr>
<td>5.4</td>
<td>Computational time results</td>
</tr>
<tr>
<td>5.5</td>
<td>Dynamical impact of BPTT learning: network stabilization</td>
</tr>
<tr>
<td>5.6</td>
<td>This figure shows how bifurcations prevents any convergence.</td>
</tr>
<tr>
<td>5.7</td>
<td>Influence of the learning algorithm on the weight distribution</td>
</tr>
<tr>
<td>5.8</td>
<td>Example of a weight distribution after an easy learning task</td>
</tr>
<tr>
<td>5.9</td>
<td>Example of weight distributions after a difficult learning task</td>
</tr>
<tr>
<td>6.1</td>
<td>Illustration of the model and the learning task</td>
</tr>
<tr>
<td>6.2</td>
<td>Capacities results on auto-associative memories</td>
</tr>
<tr>
<td>6.3</td>
<td>Content addressability results on hetero-associative memories</td>
</tr>
<tr>
<td>6.4</td>
<td>Illustration of an hetero-associative data set</td>
</tr>
<tr>
<td>6.5</td>
<td>Capacities when learning sequences of patterns</td>
</tr>
<tr>
<td>6.6</td>
<td>Content addressability results on limit cycles attractors</td>
</tr>
<tr>
<td>6.7</td>
<td>Illustration of a data set made of sequences</td>
</tr>
<tr>
<td>7.1</td>
<td>Illustration of the model and the learning task</td>
</tr>
<tr>
<td>7.2</td>
<td>Illustration of the importance of strengthening the halting test</td>
</tr>
<tr>
<td>7.3</td>
<td>Weight distribution after Hebbian learning of static patterns</td>
</tr>
<tr>
<td>7.4</td>
<td>Weight distribution after Hebbian learning one cycle</td>
</tr>
<tr>
<td>7.5</td>
<td>Dynamical properties of networks which learn static patterns</td>
</tr>
<tr>
<td>7.6</td>
<td>Dynamical properties of networks which learn sequences of patterns</td>
</tr>
<tr>
<td>7.7</td>
<td>Dynamical properties of networks which learn one sequence</td>
</tr>
<tr>
<td>7.8</td>
<td>Hebbian road to chaos: learning cycles in RNNs</td>
</tr>
<tr>
<td>7.9</td>
<td>Qualitative description of different kinds of chaos encountered</td>
</tr>
<tr>
<td>8.1</td>
<td>Capacities results, comparison with the supervised algorithm</td>
</tr>
<tr>
<td>8.2</td>
<td>Content addressability results, comparison with the supervised algorithm</td>
</tr>
<tr>
<td>8.3</td>
<td>Examples of weight distribution</td>
</tr>
<tr>
<td>8.4</td>
<td>Learning impact on the complexity of the global dynamics</td>
</tr>
<tr>
<td>8.5</td>
<td>Apparition of chaotic dynamics in between meaningful stimuli</td>
</tr>
<tr>
<td>9.1</td>
<td>Presence of nearby limit cycles in a frustrated chaos</td>
</tr>
<tr>
<td>A.1</td>
<td>Main features of the ndk: UML use case diagram</td>
</tr>
<tr>
<td>A.2</td>
<td>Main elements of the ndk: UML class diagram</td>
</tr>
</tbody>
</table>
A.3 Compositon graph between a sub-net and a neuron ............... 152
A.4 Simple network simulation using the ndk: UML sequence diagram ... 152
A.5 View of the XNDK ........................................... 154
A.6 View of the XNDK ........................................... 154
A.7 View of the XNDK ........................................... 155
A.8 View of the XNDK ........................................... 155
Chapter 1

Introduction

Most scientific discoveries and engineering advancements have come from observing the multiple phenomena occurring in Nature. From the beginning of mankind, there has been a desire to build a man or an artificial intelligence (the jew golem, the pygmalion, etc.) based on these observations. The first effective realization of a machine performing “intelligent” human tasks is often attributed to Pascal, who in 1642 created the first mechanical digital calculating machine. For the first time, a machine was able to perform tasks that people would not have believed a computer could do. It raised, for the first time, the question of comparing competence vs intelligence. Nowadays, computers can do many wonderful things. They can perform calculations millions or billions of times faster than human beings. It goes without saying that modern digital computers are capable of much more than just crunching numbers. They are able to play chess, or prove theorems. Equally there are programs that can hold a regular conversation with humans, understand stories and perform many other human-like tasks.

All these “intelligent” applications operate by manipulating symbols. They are the result of the traditional view of artificial intelligence, also known as symbolic artificial intelligence: i.e. intelligence is the result of the symbolic manipulation of abstract concepts. According to this view, the brains are symbolic processing machines. In 1980, the philosopher John Searle described a thought experiment, known as the “Chinese room argument”, in which an English-speaking person not having any knowledge of the Chinese language would sit in a room in which he/she would receive messages written in Chinese (Searle, 1984). Detailed scripts described what responses to provide. Even though the person in the “Chinese Room” may have been able to convince the Chinese interrogator that someone in the room understands Chinese, all that really happened was symbol manipulation. Searle claimed that all computers ever do or can do is manipulate symbols, without having any real understanding of what those symbols mean. According to Searle, while human minds have mental contents (semantics), computer programs are formal (syntactic), and syntax by itself is neither constitutive of nor sufficient for semantics. 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, i.e. the idea that intelligent machines will eventually possess consciousness, self-awareness and emotion.

A more recent critic of Searle states that the main difference between the brains and computers lies in a missing principle of causality (Searle, 1992). According to this view, while “brains cause minds”, this relation of causality does not exist between the outputs of a computer and its constitutive silicon chips which could be replaced by anything else without fundamental changes. The problem lies in the architecture. Most modern computers rely on an architecture where the business logic is driven by an external processing unit while the brain does not seem to make a distinction between the “processing unit” and the “business unit”.

In the eighties, these critics of the symbolic AI and top-down approach resulted in the creation of the connectionist and bottom-up approach. They in turn created the neural network architecture, based on the architecture of neurons, synapses and dendrites in the brains. Different kinds of neural nets architecture have been proposed. The family of architectures which has given the most concluding results, i.e. the feedforward neural nets, is surprisingly enough also the less plausible biologically. In feedforward nets, data is propagated linearly from input to output. In contrast, the biologically more plausible recurrent neural networks (RNNs) family, having bi-directional data flows, has still not found convincing applications. The present dissertation is part of the current connectionist effort to gain a better understanding of the non-linear dynamical phenomena occurring in fully recurrent neural networks in view of engineering and/or cognitivist applications.

The neural net computer simulations which will be presented here are based on the acceptance of a set of assumptions that for the last twenty years have been expressed in the fields of information processing, neurophysiology and cognitive sciences. First of all, neural networks and their dynamical behaviors in terms of attractors is the natural way adopted by the brain to encode information. Any information item to be stored in the neural net should be coded in some way or another in one of the dynamical attractors of the brain and retrieved by stimulating the net 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 to perturbation together with their generalization capability in the presence of variable indexing inputs. Since Grossberg and Hopfield precursor works (Kohonen, 1982), (Grossberg, 1992), the privileged regime to code information has been fixed point attractors. However, many neurophysiological reports (Nicolis and Tsuda, 1985), (Babloyantz and Lourenço, 1994), (Skarda and Freeman, 1987), (Rodriguez et al., 1999) tend to indicate that brain dynamics is much more “dynamical” than fixed points and is more exactly characterized by cyclic and weak chaotic regimes.

Many theoretical and experimental works have shown and discussed the poor storing capacity of the net when only fixed points are exploited as a severe limitation (Amit et al., 1985) and (Domany et al., 1995) for a review). It is obvious that the extension of encoding attractors to cycles potentially boosts 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:

\[(+1, +1)(+1, -1)\]  \[(+1, +1)(-1, +1)\]  \[(+1, +1)(-1, -1)\]
\[ (+1, -1)(+1, +1) \]  \[ (+1, -1)(-1, -1) \]  \[ (-1, +1)(-1, -1) \]

By studying small size fully connected networks, the first part of this thesis introduces the symbolic quantization of the network necessary to exploit these networks to encode data on their cycles. Using this mechanism, it shows how synaptic matrix randomly generated allows the exploitation of a huge number of static and above all cyclic attractors for information encoding.

A second quasi-unanimous view shared by neural net researchers is to base the learning of the synaptic matrix on a local Hebbian mechanism. The information to be stored is either installed by means of a supervised practice or discovered on the spot by an unsupervised version, revealing some statistical regularities in the data presented to the net. In this thesis, after a short review of a gradient based learning algorithm, the learning of the information to be stored will first rely on a supervised then on an unsupervised Hebbian mechanism. In the second case, we show how the storing capacity can be increased even more by leaving the semantics of the attractors to be associated with the feeding data unprescribed.

A much more questionable assumption among the scientists interested in neural networks is the presence of chaos and the benefit gained by its presence. So far the important impact of chaos in physics and the observation of chaotic regimes in EEG has not resulted in convincing practical or cognitive utilities yet. Some shy attempts have been made, where chaos is viewed as a more natural way to generate randomness or to encrypt information. This has had little impact, at least when considering neural networks as possible information storing devices. Nevertheless, chaos plays a key role in the ideas discussed in this thesis. We do, however, see chaos more as a neutral outcome of the learning practice adopted, rather than a fundamental actor whose utility can be easily identified and exploited. The more information is to be stored as attractors of the neural network, the more chaos on the road becomes unavoidable as the background dynamical regime of the net.

Ever since the seminal paper of Skarda and Freeman dedicated to chaos in rabbit’s brains (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), (Pasemann, 2002), (Kaneko and Tsuda, 2003). Chaos, although very simply produced, 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 coming stimulus. This thesis maintains this line of thinking by forcing the coding of information in robust cyclic at-
tractors and by experimentally showing that the more information is to be stored, the more chaos appears as a regime in the back, erratically itinerating among brief appearances of these attractors. Chaos does not appear to be the cause but the consequence of the learning. However, it appears as an helpful consequence that widens the net’s encoding capacity.

This thesis is built up as follows. First, in Chapter 2, different preliminary notions are reviewed for a better understanding of this thesis. In this chapter, first an introduction to neurophysiology and neurodynamics is given in order to clarify the underlying assumption of this thesis. Then, a short review of the vast field of connectionism is given to legitimate the choice of the neural net used. Finally, we provide an introduction to the dynamical system theory and to the numeral tools used all along the thesis. Chapter 3 compares our approach with other similar approaches. Moreover, this chapter highlights the meaning of “information” in this thesis, presents the very simple and classical recurrent neural network used and gives a clarification on the methodology developed.

In Chapter 4, encoding capacities and other statistical analyses are performed on random networks. Here is shown how these networks could be exploited to encode information through cycles of the underlaying dynamical system.

A first attempt to learn the information in RNNs is made in Chapter 5, namely by using a gradient based learning algorithm. Poor results are obtained. It appears that this global mechanism has difficulties to overcome the dynamics’ numerous bifurcations.

A more concluding attempt to code information in RNNs is presented in Chapter 6. This chapter describes, implements and tests the iterative supervised Hebbian learning that allows the encoding and retrieving of information in the net’s fixed point and cyclic attractors. An essential improvement of this algorithm consists of indexing the “attractor information items” by means of external stimuli rather than by using initial conditions as originally proposed by Hopfield (still used today). This addition enables the storing of cycles sharing common patterns, as introduced above for the two binary neurons example. It also enhances the content-addressability of the learned patterns and allows the exploitation of the net in a hetero-associative way.

Chapter 7 adapts this supervised iterative Hebbian learning algorithm to continuous activation function neurons, in order to enable dynamical analyses on the learned networks. Considerable differences will appear, depending on the learning task: if the storage of static patterns stabilizes the network, the coding of information in robust cyclic attractors increases the network’s chaos and the more chaos appears as a regime in the back, erratically itinerating among brief appearances of the learned attractors. When the amount of information encoded in the network goes beyond a critical value, the network turns out to be fully chaotic, with a chaotic dynamics similar to white noise.

Supervised learning has always raised serious problems both at a biological level, due to its top-down nature, and at a cognitive level. Who would take responsibility to look inside the brain of the learner, i.e. to decide on the information to be associ-
ated with the external stimulus and to exert the supervision during the learning task? As a consequence, Chapter 8 proposes an unsupervised version of the Hebbian learning mechanism where the network is taught to associate any external stimulus with a new original attractor, not specified a priori. In this sense, the network is responsible for generating the information by itself. This perspective remains in line with a very old philosophical conviction called constructivism and was modernized in neural net terms by several authors (among others (Varela et al., 1991), (Erdi, 1996), (Tsuda, 2001)). One operational form has achieved great popularity as a neural net implementation of statistical clustering algorithms (Kohonen, 1982), (Grossberg, 1992). We show that the further relaxation during learning leads to increased storing capacity and to maintaining more structure in the background chaotic regimes. In Chapter 9, the nature of these chaotic regimes is more finely analyzed, by connecting them with the classical chaotic itinerancy (Kaneko and Tsuda, 2003) and its extension to biological networks, originally called “frustrated chaos” (Bersini and Calenbuhr, 1997). The thesis will be closed by discussing the perspectives opened by this work.

All experiments and results to be discovered in the following chapters have to be assessed at the crossroad of two basic lines of research: increasing the storing capacity of recurrent neural networks as much as possible and observing and studying how this increase impacts the dynamical regimes proposed by the net in order to allow such a huge storing.

Publications


Chapter 2

Preliminary Notions

1 Introduction

The aim of this chapter is to familiarize the reader with the multiple notions, concepts and theories, which underlie our research.

Since this thesis takes its roots from neurophysiological, and more particularly from neuro-dynamical observations, the two following sections provide a short introduction to these subjects. Then, since the objects of this thesis are the computational counterparts of biological neural networks, this is followed by the connectionist neural networks, being networks made of simple interconnected units (the formal neurons). In order to legitimate the choice of the distinctive models used and the methodology developed, a wide overview of connectionist networks is provided in the fourth section. Finally, the last part of this chapter introduces the non-linear dynamical system theory, and more particularly the theories of bifurcation and chaos. This last section also describes the numerous dynamical tools used in this thesis to analyze the dynamical properties of connectionist networks.

2 Neurophysiology

Mainly two references have contributed to the writing of this section. The first one is the classic textbook entitled: “Molecular biology of the Cell” (Alberts et al., 1994), which describes life from the level of biochemistry up to the cellular level, in an integrative approach. The second reference is a book which aims at describing the PDP++ software (O’Reilly and Munakata, 2000). This software is a neural-network simulation system representing the next generation of the famous PDP software (Rumelhart et al., 1986). Quoting the authors, the goal of PDP++ is to understand how the brain embodies the mind by using biologically based computational models comprised of networks of neuron-like units. For this purpose, the authors provide a nice introduction to the neurophysiological field. The other references which have contributed to the writing of this section will be cited where appropriate.

Evolved animals are characterized by a large amount of cells (ranging from a thou-
sand to hundreds of billions). They move through their environment to feed themselves, to avoid being eaten, and to reproduce themselves. These actions are orchestrated by a complex network of specialized cells that grow long threads from their cell bodies (the nerve cells), providing rapid communication within the collective to coordinate and control the motor system. In addition, environmental physical data are transmitted to this network through sensorial channels. This defines the two roles of the nervous system: receive and transmit the sensorial information, then process and transmit the driving information.

In order to maintain its physical integrity and fulfill its main goals of survival and reproduction, animals have to take into account their environment and plan strategies of action. Thanks to the relative permanence of this environment, adaptation is made possible where specific stimuli are associated to specific actions through a mechanism of associative learning.

An increasing number of neurons are needed when the body size and/or the organism’s mobility increase. The existence of a locus, aiming to centralize and integrate neural stimuli as a whole, increases this complex neural network’s efficiency. The development of such loci is associated in evolution with cephalization, the development of a head with the accumulation of nearby sensory organs at the front end of the organism. This central core is known as the brain and consists of a great number of nerve cells and glia cells. In some developed organisms, such as human beings or dolphins, it reaches extreme complexity numbering up to 10 billions of neurons.

As all the data are being centralized in one location, it enables the brain to sort these data as a collective assuming some criterion of relevance. Other functions of the brain are to plan actions, and begin appropriate - and coordinated - motor outputs.

Normal development of the brain depends on the critical interaction between genetic inheritance and environmental experience. The genome provides the general structure of the central nervous system, and nervous system activity and sensory stimulation provide the means by which the system is fine-tuned and made operational.

While neuroanatomy studies the anatomy of neurons and the nervous system, neurophysiology studies the physical and chemical processes happening between them.

### 2.1 Elementary notions of neuroanatomy

In vertebrates the nervous system is divided into two main components: the central nervous system, and the peripheral nervous system. The central nervous system consists of the brain performing complex integrative functions and controlling voluntary activities, and the spinal cord conveying information to and from the brain. The spinal cord performs also less complex integrative functions, and directs many simple involuntary activities. These central components are highly protected by the skull and the spinal column. The peripheral nervous system consists of the nerves, some of which gather information while others transmit orders. The facial nerves enter and leave the brain directly through the skull, while other nerves reach the brain via the spinal cord. The nerves in the peripheral nervous system themselves are again divided into two
categories: the somatic nervous system and the autonomic nervous system.

The somatic nervous system is composed of nerves which play an interfacing role between organisms and their external environment, by sending information to the brain from the body’s various sensory detectors, and allowing reactions to external stimuli. The autonomic nervous system is composed of nerves which are more involved in regulating vital internal functions. They help to maintain the internal equilibrium by coordinating such activities as digestion, respiration, blood circulation, excretion, and the secretion of hormones.

2.1.1 Innateness versus Adaptation

In the past some authors have argued that the whole information contained in the brain could be provided by the genome. Among others, following a former student of Sperry:\footnote{nobel prize 1981}

“...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.” (Gazzaniga, 1992)

According to recent quantitative results, the human brain may contain as many as \(10^{11}\) neurons, each of them connected to another 10000 neurons in average, totalling approximately \(10^{15}\) synapses. Such complexity is far away from any imaginable computer simulation.

Some attempts to quantify the information complexity of the human genome assign a bit of information for every base pair. In this simplistic approach, the human genome has only about 3.5 billion bits of information. On this basis, some neural and molecular scientists have concluded that our 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 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 preestablished code! The differential expression of genes cannot alone explain the extreme diversity and specificity of connections between neurons.” (Changeux, 1985)

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

Therefore, intelligence might partly be due to adaptive rules. Based on these facts, hopes of creating an “intelligent artificial neural network” have arisen. The challenge, however, consists in finding the “right rule” enabling the machine to expand its intelligence through its past experiences.
2.2 The nerve cell: the Neuron

The neuron is the functioning unit of the nervous system specialized in receiving, integrating, and transmitting information.

Neurons are distinguished and categorized according to their general functions. More specifically there are receptor or sensory neurons, motor neurons and interneurons. Sensory or afferent (carrying toward the brain) neurons are specialized to be sensitive to a particular physical stimulation such as light (vision), sound (audition), chemical (olfaction), or pressure (touch). Motor or efferent (carrying away from the brain) neurons receive impulses from other neurons and transmit this information to muscles or glands. Interneurons or intrinsic neurons form the largest group in the nervous system. They form connections between themselves and the sensory neurons before transmission of control to motor neurons. Neurons are also classified on a morphological basis. For example, some have more elaborate dendritic branching while others differ in the placement of the cell body relative to other portions of the neuron.

While variable in size and shape, all nerve cells have four morphologically defined subunits: the dendrites, the soma, the axon and the presynaptic terminals (as shown in Figure 2.1). The cell body, or soma, is the neuron’s main cellular space. The soma houses the nucleus, in which the neuron’s main genetic information can be found. The dendrites are short, highly branched fibers, that receive messages (signals) from other neurons and carry them toward the neuron’s cell body. The axon is a long fiber (which can extend up to 1m) that carries signals away from the cell body. Towards its tip, or tips, the axon splits into a tree. The tips of the axon, called the presynaptic terminals, come into close contact with the dendrites of other neurons or with muscles. The number of presynaptic terminals of one neuron can be huge (up to 10000). By means of its terminals one neuron transmits information about its own activity to receiving structures (e.g. dendrites of other neurons).

2.2.1 Transmission of the nerve impulse

Two means of propagation intervene into the transmission of information from one neuron to another. The first one is the propagation of an electrical signal, called an action potential, which travels down the axon. When the action potential reaches the presynaptic terminals, information is propagated through specific signalling chemicals, the neurotransmitters, that cross the synaptic gap and propagate this signal to the receiving structure. This second mechanism of propagation is also called synaptic transmission.

The plasma membrane of neurons, like all other cells, has an unequal distribution of charged ions between the two sides of the membrane. The outside of the membrane has a positive charge, while the inside has a negative charge (at rest, the membrane potential is about $-70\, mV$, see Figure 2.1). Ion concentrations inside the cell are controlled by the opening of ion channels, under the control of neurotransmitters.

An action potential is created and emitted through a process of depolarization. The depolarization wave is propagated and when it reaches the presynaptic termi-
Figure 2.1: Basic description of a neuron, its action potential and the neurotransmitters involved in the transmission of the nerve impulse to other nerves or muscles.

nals, chemicals called neurotransmitters are released and travel across the gap (the synapse) between the terminal and the dendrite of the neighboring neuron. These chemicals are used to communicate the signal from one cell to the next by opening or closing ion channels. Some neurotransmitters are excitatory and depolarize the next cell, increasing the probability that an action potential will be fired. Others are inhibitory, and cause the membrane of the next cell to hyperpolarize, thus decreasing the probability of the next neuron firing an action potential.2

2.2.2 Neuronal Adaptation

Learning mechanisms involve ongoing adaptations of the brain throughout its lifetime in response to its environment. In order to explain how experiences of various kinds produce short- and long-lasting changes in the function and structure of the nervous system, different forms of neuronal adaptation have been proposed.

One of the major goals of modern neuroscience research is to identify and explain these neuronal changes which are likely to be the physical basis of learning and memory. A typical way to classify among these various forms of neuronal adaptation is through their permanence:

**growth and death of synapses, neurons and myelin sheaths** From the embryo stage

---

2The fact that our brain is not a “solid,” all-electrical continuous network, and instead needs biochemical changes at crucial points for a signal to travel across it, is the foundation of learning. It also explains how by modifying biochemical reactions, different factors like drug or alcohol consumption can perturb our perceptions. Psychotropic drugs often act precisely in the same way as these neurotransmitters.
until early childhood, there is an intense proliferation of the nerve cells. Then, there is a massive extinction of synapses and neurons hereby achieving a high level of organization by the definition of communication pathways between neural groups and by enabling these neural groups to specialize. This extinction occurs through environmental interactions and enables each individual’s nervous system to be precisely optimized to take account of physical differences in the body’s geometry—for example, the distance apart of the eyes (Bland, 1998). Once this scaffold is set, it will remain nearly identical all along the individual’s lifetime. Other hypotheses exist. For example, if Changeux agrees that all adaptive brain changes, occurring between birth and puberty in humans, involve the elimination of preexisting synapses, he states that these preexisting synapses were not necessarily all present from the beginning. From birth to puberty, Changeux hypothesized that waves of synaptic growth occur, with subsequent experiences serving to retain the useful ones while eliminating the useless and redundant ones (Changeux, 1985). This learning mechanism has given rise to numerous learning algorithms in connectionist models. Most of these algorithms, being evolutionary algorithms, rely on a kind of randomness at one level or another.

long term adaptation  This kind of adaptation enables durable changes (from hours to years). It is generally accepted that no architectural (structural) neural modifications occur at this level. This adaptation is based only on modifications of synaptic efficiencies (called “synaptic plasticity”). The mechanism of long term potentiation, described in the next subsection, has been identified as one of the mechanisms responsible for this neural adaptation. In the connectionist field, it has resulted in the large family of Hebbian learning algorithms.

short time adaptation  This kind of adaptation provides the basis for rapid behavioral changes based on context modifications. The underlying hypothesis directing some connectionist approaches (and ours among others) is that this kind of neuronal adaptation is based on dynamical processes and therefore is 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.

2.2.3 Synaptic plasticity

The Spanish anatomist Santiago Ramon y Cajal was among the first neuroscientists to suggest that learning was not only the product of new cell growth. In 1894 he suggested that memories might be formed by strengthening the connections between existing neurons to improve the effectiveness of their communication. Hebbian theory, introduced by Donald Hebb in 1949, echoed Ramon y Cajal’s ideas, and further proposed that cells may grow new connections between each other to enhance their ability to communicate:
“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.” (Hebb, 1949)

Such adaptation is generally thought to occur through long-lasting changes in synaptic strength, i.e. the facility with which a pre-synaptic neuron influences the potential of a post-synaptic neuron.

This intuition was experimentally confirmed in 1973 by the discovery of the long-term potentiation (LTP) effect in the hippocampus of rats (Bliss and Lomo, 1973). The authors observed an increase in synaptic efficacy lasting from hours to days following brief tetanic (high-frequency) electrical stimulation of an afferent pathway. It proved that modifications in the strength of synaptic connections can contribute to the substratum for learning.

The original version of the Hebbian rule explains how a phenomenon of mutual excitation strengthens the connections. A natural extension of this rule is to decrease the synaptic strength when the source and target neurons are not active at the same time. It takes into account the fact that memories may be forgotten through the weakening or loss of connections\(^3\). This effect is due to a mechanism of long-term depression (LTD), opposite to the LTP (Kirkwood and Bear, 1994). LTD is produced by nerve impulses reaching the synapses at very low frequencies. The synapses then undergo the reverse transformation from LTP: instead of becoming more efficient, the synaptic connections are weakened.

The generality and the presence of these two mechanisms among most of the synapses make us assume that they play an important role in the formation and elimination of some types of memories. In fact, they are still the most widely studied cellular models of synaptic plasticity. Elucidation of their underlying molecular mechanisms may offer important insights into processes underlying learning and memory.

In the connectionist field, the mechanism of Hebbian synaptic plasticity has led to numerous learning algorithms. Chapters 6, 7 and 8 will adapt and then propose new implementations of these algorithms.

\(^3\)For example, a man might be startled by the sound of a car alarm outside. Sensory cells in the ear record the sound and send it to the brain where it activates neurons that control the man’s muscles. But as the blaring alarm continues, those connections are weakened so that the alarm no longer causes the man to be startled.
3 Neurodynamics 
and the physiological basis of behavior

The brain is a physiochemical system that operates simultaneously at many hierarchical levels. If it produces intelligent behavior, the crucial question for neuroscientists is to find which level of functioning is relevant for explaining this behavior. Neuroscientists seeking to discover the physiological basis of behavior have located it at various levels of this hierarchy.

3.1 The neuron doctrine

The first, and still actual, main tendency located the physiological basis of behavior at the level of individual neurons. This view, known as the "neuron doctrine" (Barlow, 1972) is mainly based on Hubel and Wiesel's discovery of cells in the visual cortex that respond to particular features presented at specific locations in the visual field. Some cells were found to respond to simple features such as a local edge having a particular location and orientation, while other cells had more complex response functions, as if in response to spatial combinations of simple cell responses (Hubel and Wiesel, 1962).

Other researchers believe that the explanation of behavior should be sought in phenomena at much lower levels of description. For example, researchers have designated biochemical changes at the synapse as the biological basis of behavior. Others, such as Penrose and Hameroff (Hameroff and Penrose, 1996) locate it at the level of the microtubules, nanosized tubules running through our neurons, which act as part of a quantum computer.

3.2 The connectionist approach

A different approach has been taken by researchers who postulate that the changes involved in learned behavior, although based on or involving cellular and molecular modifications, are widely distributed spatially and should be understood first at the level of the neural network (among others (Rosenblatt, 1962), (Rumelhart and McClelland, 1986), (Sejnowski and Rosenberg, 1987), (Hopfield, 1982), (Amari, 1983), (Elman, 1991)).

This view, called "connectionism", has led to the development of simplified models of the brain as being composed of large numbers of inter-connected units (the analogs of neurons) with weights that measure the strength of connections between the units.

---

4 Neuroscience can be interpreted in different ways. First, it can be understood as the science called biological neuroscience, concerned with the investigation of the structure and function of individual neurons, neuronal ensembles and neuronal structures. According to another conception (used here), neuroscience is taken to be what is often called cognitive neuroscience. Cognitive neuroscience is an interdisciplinary approach to the study of the mind, the goal of which is the integration of biological and physical sciences - included in particular biological neuroscience - with the psychological sciences to provide an explanation of mental phenomena.
Learning takes place by strengthening and weakening the connection strengths between the units in the network in parallel.

Two connectionist trends coexist. The first one, typified by so-called PDP models relies on feed forward networks, the other is characterized by self-organizing dynamical systems and relies on recurrent networks. However, both of them still assume that the biological basis of behavior can be explained in terms of the properties of the system’s parts and that neural functioning must be seen as a passive reaction to the stimulus. The foundation and the explanations of these neural networks remain at the level of the action potential of single neurons.

### 3.3 The dynamical approach

Until recently, the knowledge of the brain was principally anatomic. Different brain areas, each of them being associated to specific functionalities, have been identified from brain lesion observations. However, such studies did not provide information on the possible relations between these different areas. The vision of the brain was static and compartmental (in a continuation of the phrenologist view).

The development of modern investigation tools has enabled dynamical observations of the brain activity and has suggested the explanation of behavior in the light of dynamical brain state. Several tools exist, among others, the multielectrode electroencephalogram (EEG) and magnetoencephalogram (MEG), functional magnetic resonance imaging (fMRI) and scanning by positron emission tomography (PET-scanners). Quoting Freeman:

“Nox, 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” (Freeman, 2002).

---

5 In the phrenologist view, each distinct faculty of the mind is associated with a separate seat or “organ” in the brain. Furthermore it was believed that the size of an organ measures its power. Thus, since the shape of the skull is related to the shape of the brain, the surface of the skull can be read as an accurate index of psychological aptitudes and tendencies. (Letter from Dr. Gall, to von Retzer: “upon the Functions of the Brain, in Man and Animals”, 1798)
And in another paper:

“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” (Skarda and Freeman, 1990b).

These dynamical observations show that while the activity of single cells appears to be largely unpredictable and noisy, the mass of cells cooperates to produce a coherent pattern that can be reliably related to a particular stimulus.

The leading experiment of this dynamical paradigm using EEG is usually attributed to Skarda and Freeman. The method developed to obtain reliable scientific data is based on simultaneous EEG recordings among different cortical areas during cognitive tasks. In their seminal, particularly demonstrative experiment, they analyzed the olfactory bulb of the rabbit and 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. More specifically, they have shown that natural attentive waiting states correspond to chaotic dynamics, and that the presentation of a known odor leads, through a bifurcation, to almost cyclic dynamics (Skarda and Freeman, 1987). Even local and specific to the olfactory bulb, these results may indicate the relevance of taking inspiration from the dynamical systems theory for the analysis of brain processing. Consequently, ever since, a large amount of research has been dedicated to the dynamical analysis of the brain (as indicated by the increasing number of publications as well as scientific conferences).

These observations have given rise to what has been called the “Dynamical Hypothesis” according to which, such global dynamical behavior of brain areas is of paramount importance for its understanding. A detailed introduction to this field with related papers is given in (Guillot and Dauce, 2002). However, the results’ func-

---

6“...In the experiment, thirsty rabbits were conditioned to lick in response to an odorant, followed after two seconds by delivery of water, and just to sniff in response to an unreinforced odorant. Recordings of 64 EEG potentials were made simultaneously. The typical pattern of the bulbar EEG was a slow wave, called a respiratory wave, with a burst of oscillation in the gamma range (35–90 Hz) common to all channels. Analysis revealed that odor specific information existed in spatial patterns of amplitude of the oscillatory burst. Analysis of the EEG traces showed that in the background before conditioning, every trace had the same temporal waveform, but that the amplitude differed between the channels forming a relatively constant spatial pattern that could be easily identified with a particular animal and that remained constant until odorant conditioning was undertaken. No changes in this background pattern occurred when unreinforced odorants were presented to the animal; however, new patterns did emerge with reinforced odorants. These patterns remained stable within and across sessions provided the stimulus–response contingencies were not changed. Of particular interest is the fact that these patterns were globally distributed in the bulb.”
tonal significance remains a matter of debate. According to Freeman:

“Most neuroscientists of the “neuron doctrine” reject EEG and MEG evidence, 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” (Freeman, 2000).

Two types of observations are usually distinguished from dynamical observations. The first one being the phenomenon of synchronization occurring between neuronal groups and the second one being the nearly total presence of chaotic dynamics.

### 3.3.1 Synchronization and coherence of neuronal activity

In physics, synchronization of coupled oscillating systems is the appearance of certain relations between their phases and frequencies. This definition is directly applicable in neurophysiology: “synchrony” or “phase synchrony” indicates neuronal groups that oscillate into distinct frequency bands (delta: 1–4 Hz, theta: 4–8 Hz, alpha: 8–12 Hz, beta: 12–30 Hz, gamma: 30–80 Hz, also referred to as 40 Hz) that enter into precise phase-locking over a limited period of time (Nunez, 1981).

The results of several experiments in animals exhibit a robust correlation between behavioral states and transient periods of synchronization of oscillating neuronal discharges in the frequency range of gamma oscillations (Skarda and Freeman, 1987), (Gray et al., 1989), (Neuenschwander et al., 1996). More recent but similar results have been obtained for humans during visual cognitive tasks while facing ambiguous visual stimuli: synchronization appeared to be responsible for the binding of different but related visual features so that a visual pattern can be recognized as a whole (Rodriguez et al., 1999).

Many authors have suggested that these periods of synchronization act as a central integrative mechanism that brings a widely distributed set of neurons together into a coherent ensemble that underlies a cognitive act. Following the same idea, the desynchronization would reflect a process of active uncoupling of the underlying neural ensembles necessary to proceed from one cognitive state to another (Skarda and Freeman, 1990a), (Varela, 1995).

### 3.3.2 Chaos as the basal state of neuronal activity

Coherence analysis of the EEG among different cortical areas sheds a light on the different states of the brain. For example, in deep anesthesia, coma, or brain death, the brain is “attracted” to rest (it returns to a rest state after being stimulated). Therefore, it has an equilibrium steady state “attractor”. The phenomenon of epilepsy is correlated to an abnormal phase synchrony among the entire brain, indicated by the presence of a limit cycle attractor.

During awake activity, according to EEG observations, brain activity is typically aperiodic and unpredictable in the absence of stimulation, and it returns to the same following termination of a stimulus. These observations have led several authors
(among others (Nicolis and Tsuda, 1985), (Babloyantz and Destexhe, 1986), (Skarda and Freeman, 1987)) to conclude that brain activity has chaotic deterministic dynamics. Furthermore, during cognitive tasks (or when recognized stimuli are presented to the brain), the brain usually enters an “almost cyclic” dynamics.

These observations have suggested the incorporation of chaotic dynamics into connectionist models (Sompolinsky et al., 1988) with the implication that chaos may provide the basis for flexibility, adaptiveness, and the trial–and–error coping that enables the nervous system’s interaction with an unpredictable and ever-changing environment. Our approach follows the same idea: dynamics occurring in small size and very simple (compared to the brain) connectionist networks are being studied. Limit cycle attractors will be used to carry meaningful information. Chaotic dynamics will appear very often and will provide rapid adaptation to the system.

However, according to Rapp (1993), all these results have to be taken with caution. The analyses performed on EEG brain observations are based on powerful mathematical tools provided by the theory of non-linear dynamical systems. According to these analyses, chaotic dynamics appear on dendritic synaptic potentials and axonal action potentials. However, one can easily be misled by the results of algorithms indicating the presence of chaotic organization in the presence of pure noise.

3.4 Chaos, Brain and Cognition

Currently, most of the scientific community agrees that aperiodic and unpredictable signals obtained from brains EEG represent deterministic chaotic dynamics, even if many results have to be looked at with caution. The usual awake brain activity corresponds to chaotic dynamics, while epilepsy would correspond to a global diminution of the complexity (resonance appearing at the brain scale) (Cohen et al., 2002) and deeper chaotic dynamics would be characteristic to depressive state (Thomasson et al., 2000). However, a deeper chaos or a less complex dynamics corresponds to a pathological state only if it appears at the brain scale. At a local scale, when occurring in a particular neural group, it has another meaning: a limit cycle attractor would be an indicator of a cognitive process (Skarda and Freeman, 1987).

Considering chaotic dynamics as the basal state of behavior and limit cycle attractors as the signature of cognitive processes sounds appealing. In fact, these ideas have been the “thread” conducting this thesis.

However, according to Freeman:

“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.” (Freeman and Skarda, 1990).

Following a recent opinion, it seems more plausible that the brain processes information using exclusively chaotic dynamics (Skarda and Freeman, 1990a), (Tsuda, 2001).
A cognitive process would reflect the passing from one chaotic attractor to another chaotic attractor. This leads to the idea of storing information in chaotic attractors instead of in limit cycle attractors.

One of the consequences would be that memories are not to be seen as static features of the brain retrieved on demand and perfectly recovered. Memories are no longer stored in a fixed point or a limit cycle attractor which are perfectly identifiable and thus recoverable features. Instead, memories are re-created, always with differences, at the same time that they are retrieved. Hence memory becomes an active process.

If these ideas appear as quite revolutionary in the cognitivist field, they are more familiar in philosophy. In Nietzsche’s philosophy, the transvaluation of all values is an active process where continuously values must be reevaluated, reinterpreted and readapted (Nietzsche - Généalogie de la morale). Also according to Klossowski, the idea of the “eternal return of the same” is to be enlightened by an active process of forgetting. Always the same is rediscovered. And according to Nietzsche:

“You need chaos in your soul to give birth to a dancing star”

---

8 Friedrich Nietzsche: Thus Spake Zarathustra, 1891
4 Connectionist Models

Historically, connectionist models have been developed from the perspective of cognitive science. Connectionism is used to answer questions pertaining to human cognition, from perceptual processes to “higher level” processes such as attention and reasoning.

The birth of cognitive science is often traced back to the Symposium on Information Theory held in 1956 at MIT (Massachusetts Institute of Technology). There, researchers from various disciplines gathered to exchange ideas on communication and human sciences. It was the seed for cognitive science defined as the interdisciplinary study of mind:

“Three talks in particular, Miller’s The magical number seven, Chomsky’s Three models of language, and Newell and Simon’s Logic theory machine, have been singled out as instrumental in seeding the cognitive science movement. Following these talks, a perception began to emerge that human experimental psychology, theoretical linguistics, and computer simulations of cognitive processes were all pieces of a larger whole.” (Medler, 1998)

Although each discipline has its own unique interpretation of cognitive science, they are bound into a cohesive whole by a central tenet. This tenet states that the mind is an information processor. According to Pollack:

“Under an assumption that the mind arises out of the brain, a reasonable research path to the artificial mind was by simulating the brain to see what kind of mind could be created” (Pollack, 1988).

Connectionism -within cognitive science- is the name for the computer modeling approach to information processing based on the design or architecture of the brain. A connectionist model, or neural network, is, like the brain, a massively parallel collection of small and simple processing units where the interconnections form a large part of the network’s intelligence. Of course, in terms of scale, a brain is massively larger than a neural network (if the human brain has approximatively 10 billion of neurons, usually artificial neural networks have no more than hundreds of units), and the units used in a neural network are typically far simpler than real neurons. Nevertheless, certain functions that seem exclusive to the brain such as learning, have been replicated on a simpler scale, with neural networks.

Another approach treats neural nets from an engineering perspective as technological systems for complex information processing. Therefore, neural nets are evaluated according to their capacity to deal with complex problems, especially in the areas of association, classification and prediction (Haykin, 1998).
4.1 History

The first mathematical formalization of connectionist models appeared in a 1943 paper by McCulloch and Pitts (McCulloch and Pitts, 1943). After defining a formal neuron as a simple processing unit, they proved that any statement within propositional logic could be “implemented” by a network of simple processing units. In 1949, Hebb proposed a theory of behavior based as much as possible on the physiology of the nervous system, giving the connectionism a neuropsychological base. Rejecting reflexes, Hebb put forth and defended the notion of an autonomous central process, which intervenes between sensory input and motor output. Hebb is generally credited with two notions that continue to hold influence on research today. The first notion is that memory is stored in connections and that learning takes place by synaptic modification. The second being that neurons do not work alone, but may, through learning, become organized into larger configurations, or “cell-assemblies”.

In the sixties, the invention of the modern computer enhanced the development of connectionism: theories could now be stated more formally and investigated on artificial computational devices. Selfridge’s Pandemonium (Selfridge, 1959) and Rosenblatt’s perceptron (Rosenblatt, 1962) did a lot to further the concepts of connectionism. Supervised learning algorithms based on the estimation of the error between the observed output and the desired output were developed (they would lead to the backpropagation rule). However, a gap appeared between the high expectation of these models and the results obtained. Many of the early workers in this field were given too extravagant or exuberant claims or overly ambitious goals (Pollack, 1988). As a consequence, when at the end of the sixties, Minsky and Papert proved the inherent limitations of the simple perceptron (Minsky and Papert, 1969), nearly causing the complete abandoning of connectionism. During ten years, researchers could not easily publish their work in the AI journals or conferences.

The interest in connectionist modelling has increased again since the 1980’s with the development of new kinds of architecture and learning mechanisms. Hopfield built a bridge with statistical physics by defining a connectionist model for building associative memories based on an analogy to a well-studied physical system, the spin glasses (Hopfield, 1982). Another notion from physics which has been ported into connectionism is simulated annealing (Kirkpatrick et al., 1983). It defines a Boltzmann Machine for which a global minimum can be found by using a very simple learning procedure, using only local information, to interactively adjust the weights. In 1982, Kohonen developed self-organizing maps used for pattern recognition with the help of a non-supervised learning technique based on competition (Kohonen, 1982). In 1985, by modifying the activation function of the units from a binary to a continuous, analogue threshold, it was possible to adapt the error algorithm from single layer networks to multi-layer feedforward networks by using a mechanism of backpropagation of errors to adjust the weight connections (Rumelhart et al., 1986). This backpropa-

---

9 The Boltzmann machine uses a stochastic method: the probability of a unit’s next state is a function of a global parameter which defines the “temperature” $T$.

10 Continuous neurons are required since the error backpropagation is based on the deriva-
gation algorithm led the way to numerous practical applications in the fields of classi-
ification and data analysis. Another important result has been the proof that this kind
of networks with a sufficient number of hidden units, is capable of approximating any
continuous function to any desired accuracy (Cybenko, 1989).

Ever since, connectionist research has been broken up in numerous different direc-
tions. First of all, different cognitivist approaches led to different models. In addition,
models which found practical engineering applications were subject to improvements
which do not pretend to follow any biological inspiration.

4.2 Structure of Connectionist Models

Since connectionism adopts the view that the basic building block of the brain is the
neuron, the functional properties of the brain, required for information processing,
must appear in connectionist models. The functional properties of connectionist mod-
els can be summarized in three basic tenets. First, signals are processed by elementary
units (formal neurons). Second, processing units are connected in parallel with other
processing units, which defines the architecture. Third, connections between process-
ing units are weighted following some learning mechanisms. These three tenets need
to be broad in their descriptions in order to accommodate all aspects of connectionism.

4.2.1 The formal neuron

An artificial neuron is based on a very simple abstraction of real brain neurons (see
Figure 2.3). It is a functional unit giving a different output in function of the state of
its inputs and its internal state.

Basically, it simulates the basic functions of natural neurons:

1. various inputs feed the neuron;
2. each of these inputs is multiplied by a connection weight;
3. the obtained products are simply summed to elaborate the neuron’s potential.
   A threshold term is sometimes added;
4. the potential feeds a non-linear transfer function. This leads to the activation of
   the neuron which is the output value.

A crucial question to be asked is how neurons encode information in the sequence
of action potentials they emit. Different choices can be made, each one leading to
important differences. Among them the most important is the choice of the transfer
function giving different ways to code the information.

The first generation: the threshold gate

Historically, the first artificial neuron, formalized by McCulloch and Pitts, used
a threshold transfer function: all or nothing, giving discrete neuron activation. The
characteristics of their model was that they treated a neuron as a binary device, distinguishing only between the occurrence and the absence of a spike to encode the information. The threshold gate was used as the building block for various network types including multilayer perceptrons, Hopfield networks and the Boltzman machine. It turned out that the threshold gate is a computational powerful device, despite its lack of biological plausibility.

The second generation: rate of discharge

The neuron is activated using a continuous activation function which most of the time is sigmoid-like shaped, thus, a limited continuous function steadily increasing. It is thought that this model is biologically more meaningful since the continuous output of the neuron could represent the firing rate of the biological neuron. The firing rate corresponds to the number of spikes per second emitted by the neuron. It is assumed that this number could encode relevant information, especially in the primary sensory areas where low level information processing takes place.

Historically, these continuous activation functions appeared to define a more powerful learning procedure. Indeed, to back-propagate the error obtained by the Widrow-Hoff rule (Widrow and Hoff, 1960) along the multiple layers of a feedforward network, activation units must be continuous. It has been proven that these kind of networks using non-linear transfer functions can in principle compute any analog function (Cybenko, 1989), raising the problem of learning.

It has to be noted that in these two models of formal neurons there is no explicit notion of time. In the first model, time does not appear at all. When using the rate
of the discharge model, time only appears in the biological interpretation. This lack of explicit time dependence enables the usage of discrete temporality, easing the implementation of these models. Both kinds of activation functions will be used in our simulations.

The third generation: spiking neurons and integrate-and-fire models

However, although observations show that firing rates play an important role in the nervous system, further experimental results indicate that some biological neural systems use the exact timing of individual spikes, raising the idea that the firing rate alone does not carry all the relevant information.

A third generation of neurons has been introduced to take into account the timing of individual spikes (Maass and Bishop, 1998). Thus, time plays a key role in these models, resulting in a better level of definition of the temporal behavior of neurons. This kind of formal neurons has not been investigated in this thesis.

4.2.2 Architectures

Once the type of unit is chosen, all the processing units have to be connected in parallel with each other, hereby defining the architecture of the neural net (or its topology). In function of a problem to be solved or a system to be studied, a specific and adapted architecture is developed. Each unit (or group of units) receives a specific site in the architecture that defines its function (in a classification problem: input layer, output layer, associative layer, etc.). This remains the same in both the cognitivist and the engineering perspective.

A huge number of different architectures exist in literature. Also there is no absolute way to categorize them. From a topological perspective, two main types of architecture exist: feedforward and recurrent neural networks. In feedforward networks, activation is “piped” through the network from input units (neurons without incoming links) to output units (neurons without outgoing links). In contrast, recurrent neural networks (RNN) have at least one cyclic path. See Figure 2.4.

![Figure 2.4: Typical structure of a feedforward network (left) and a recurrent network (right).](image)
The feedforward network family can be divided into different subgroups in function of the learning algorithm used. Among others, the layered feedforward network family is composed of the multilayer perceptron (MLP), the radial-basis function (RBF) network, the principal component analysis (PCA) network and the self-organizing map (SOM). The common ancestor of this family is Rosenblatt's perceptron. A biological interpretation appears in the retina where an hierarchical feedforward cortical architecture is used for preprocessing visual information. These networks are characterized by the coupling of an input and an output. There is an input layer of source nodes and an output layer of neurons (i.e., computation nodes). In addition there are usually one or more layers of hidden neurons, which extract important features contained in the input data. These networks are commonly used for hetero-associative memories, classification, function approximations (it has been proven that MLPs can approximate any function with an arbitrary precision), clustering, etc. Feedforward networks by themselves are nonlinear static networks. They can be made to operate as nonlinear dynamical systems by incorporating short-term memory into their input layer. The network obtained is sometimes called the focused time-lagged feedforward network (TLFN) (e.g., the Tapped-Delay-Line (TDL) memory). An attractive characteristic of nonlinear dynamical systems built in this way is that they are inherently stable.

As opposed to feedforward networks, which only have connections between layers, recurrent networks can have synaptic connections between any pairs of neurons. Recurrent networks 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. In this way, the network's activity can at any point in time reflect whatever external input is presented to it, plus its own prior internal state. This defines dynamical systems. It has to be noted that all biological neural networks are recurrent. A variety of algorithms and architectures exist, making such recurrence possible. The pioneer work is attributed to Hopfield. The Hopfield network has no special input or output neurons, but all are both input and output (it defines an auto-associative memory), and all are connected to all others in both directions (with equal weights in the two directions). Hopfield showed that this model defined a dissipative system for which it was possible to define an energy function, always decreasing along dynamic evolutions. The shape of the energy landscape shows some local minimum (the attracting fixed points) with variable basins of attraction. Retrieving a memory from a partial input corresponds to starting somewhere high on the landscape, and rolling into the nearest minimum.

Since feedforward nets have shown good results in many practical applications in different areas - from classification to time-series prediction - a huge part of the neural net literature is dedicated to them (≈ 95%). In contrast, recurrent nets are not covered in most neuro-informatics textbooks, and are absent from engineering textbooks because of the theoretical and practical difficulties preventing their practical applications.

In this thesis, only fully recurrent neural networks will be studied. However, it is
quite obvious that cognitive processes and/or more practical applications will require higher level architectures. One of the goals of this thesis was to enable and prepare future works by developing a generic neural software platform facilitating the development and testing of any kind of neural architecture.

4.2.3 Learning Mechanism

A connectionist model is characterized by its functional unit, its architecture and the learning rule used.

The question of learning is among one of the most challenging problems in neuroscience. Basically, a learning mechanism defines a process by which the free parameters (i.e., synaptic weights and bias levels) of a neural network are adapted through a continuing process of stimulation by the environment in which the network is embedded (Haykin, 1998). The type of learning is determined by the manner in which the parameter changes take place. Different types of classifications exist. Only two of them are reviewed here:

**Supervised versus unsupervised learning rules** Following the type of data to learn, the learning may be supervised (with a teacher) or unsupervised (without a teacher).

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 free parameters so that its actual output due to a given input is in a statistical sense close enough to the desired output for all the inputs. For example, the mean-square error may be used as the index of performance to be minimized.

Unsupervised learning procedures do not rely on specified output data, no teacher is needed to define the correct output. The adjustment of synaptic weights may be carried through the use of neurobiological principles.

**Local versus global learning rules** It is possible to differentiate between a local or a global scale learning process. The former relates to synaptic processes, the latter to unspecific modulatory systems;

A learning rule is said to be local if the update of a particular connection depends only on information available to the neurons on either side of the connection. The most known is the Hebb rule which suggests that neural pathways are strengthened each time they are used. Locality is important because it provides a natural parallelism to the implemented algorithm.

Global rules take into account the global characteristics of the network’s activity. Thus, the update of a particular connection relies on information which is not available locally to the neuron (for example a comparison with its neighbors, or with an ideal output). Biologically these rules seem less plausible.

Examples of learning algorithms:
Self-Organizing Maps (SOM) learning provides a global and unsupervised learning process. It defines a mapping from an input signal of arbitrary dimension to a one or two dimensional array of nodes which corresponds to a discrete map. The biological basis of SOMs is that sensory inputs, such as motor, visual, auditory etc., are mapped into corresponding areas of the cerebral cortex in an orderly fashion. Thus, this method converts complex, nonlinear statistical relationships between high-dimensional data items into simple geometric relationships on a low-dimensional display. It thereby compresses information while preserving the most important topological and metric relationships of the primary data items on the display. Therefore it may also be thought to produce some kind of abstraction, and hence can be used in data mining and clustering.

The backpropagation algorithm provides a global and supervised learning process. This algorithm is usually applied to multilayer perceptron or radial basis functions networks. It consists of backpropagating the error between a desired output and the output obtained from the last layer to the first layer in order to adjust the weight connections. This rule is biologically not plausible. However, it has enabled numerous practical applications in many fields such as classification, time-series forecasting, function approximation, etc.

The backpropagation algorithm can also be applied to recurrent networks by unrolling these networks through time. This gives the backpropagation through time (BPTT) algorithm, which is used in time-series forecasting and prediction. This learning rule will be fully described in Chapter 5.

In its classical version, the Hopfield model uses a local and supervised version of the Hebb rule. This learning rule will be described in detail in Chapter 6.

5  Dynamical systems theory, chaos and recurrent neural networks

Artificial neural networks are dynamical systems. They are analyzed as such in this thesis in order to show that information can be encoded using their different potential attractors. Before analyzing neural networks dynamics, we will extensively introduce the basic notions of the dynamical systems theory, with a particular emphasis on the chaos theory. We will also explain the mathematical tools available to characterize them. For a deeper view, the following literature is suggested to the reader: (Ott, 1993) and (Devaney, 1989).

5.1  History

Chaos is the supreme ideal of Taoism. Chaos is wholeness, oneness and Nature. The quote from Tchouang Tseu at the beginning of this thesis represents Chaos as the natural state of the world. Digging holes in its head means destroying the world’s nat-
ural state. Therefore, to the ancient Chinese people Chaos represents a “respectable aesthetic state”.

The above interpretation of chaos is very different from its western counterpart, where chaos more has a meaning of disorder. In the Theogony, the earliest western literature that we consider here, Hesiode describes Chaos as the first primitive god: “Verily at the first Chaos came to be,...”\textsuperscript{11} From Chaos, the primitive disorder, Gaia, the earth, the order, came into being. This has conducted first to the belief, then to the scientific paradigm of an ordered and deterministic world, exempt of any chaos, or disorder.

This thesis is based on a third notion of chaos: the deterministic chaos originating from the non-linear dynamical system theory. Though this mathematical chaos looks very different from the western primordial Chaos, it will seem very related to the old Chinese meaning.

The dynamical systems theory was made possible by Galileo’s (1564-1642) introduction of time in mathematical and physical models. At that time - and until the end of the nineteenth century - there was a deterministic view of the world. Newton (1642-1727) formalized Pascal’s idea of an explicable universe. Mathematical and physical systems were seen as reversible in time, foreseeable and reproducible. The climax of determinism was crystalized by Laplace (1749-1827):

“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 an intellect nothing could be uncertain and the future just like the past would be present before its eyes.” introduction to the “Essai”

This intellect is often referred to as Laplace’s demon. At the end of the nineteenth century and the beginning of the twentieth, uncertainties arose. Clausius (1822-1888) showed the irreversibility of chemical processes and defined the second law of thermodynamics: the irreversible increase in entropy. Boltzmann (1844-1906) explained irreversibility in physics with statistic mechanics. Quantum mechanics (1920-1930) introduced unpredictability as an intrinsic feature to matter. Godel (1930), with the Incompleteness Theorem, produced fundamental results with regard to axiomatic systems, showing that in any axiomatic mathematical system there are propositions that cannot be proven or disproven within the axioms of the system. In particular, the consistency of the axioms cannot be proven. This ended a hundred years of attempts to establish axioms which would put the whole of mathematics on an axiomatic basis.

By studying the three body problem (nine simultaneous differential equations - considered as one of the most difficult problems in mathematical physic), Henri...
Poincaré (1854 - 1912) 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)

However, it was not before the sixties that the so called “Theory of Chaos” was born. Before that scientists, intrigued by the deterministic chaos concept, were warned by their supervisors and colleagues that such research could cost their respectability, and possibly their careers (Gleick, 1987). At that time, chaos was not a science, or even a cohesive theory, but rather an untested discipline without real experts.

In 1963, Lorenz published a seminal paper describing behaviors of non-linear dynamical systems inspired by the simple modeling of the earth’s atmosphere with 3 variables (Lorenz, 1963). According to some values of the states he described the apparition of a new dynamical behavior: while the activity of the 3 variables looked erratic and unpredictable, their visualization in phase space showed the presence of an attractor finely striated, whose shape and regularity evoked the wings of a butterfly. In 1971, Ruelle and Takens called such attractor a strange attractor, after obtaining similar attractors from turbulence analyses (Ruelle and Takens, 1971). In 1975, Mandelbrot showed that their geometrical properties revealed a new structure that he called fractal (Mandelbrot, 1975). By the mid 1970s, the movement toward chaos as a science was well underway, and in 1977, the first conference on chaos theory was held at a gracious villa in lago di Como, Italy. Perhaps the most startling finding coming out of this new scientific theory was that order exists within chaos: determinism may lie in the apparently most disordered system and order appears from chaotic conditions.

### 5.2 Definition and general properties

Dynamical systems is the study of the iteration of functions from a space into itself—in discrete repetitions or in a continuous flow of time. The choice of the temporal state is determining and depends on the studied system in general. Usually continuous time systems are used to describe the evolution of physical data described by differential equations, and discrete time systems are usually used when relying on computational models. The latter will be used in this thesis.
Qualitative and quantitative descriptions of the dynamics can be given from different perspectives. One perspective consists of analyzing the statistical properties of the dynamical system. This field has been widely investigated by Cessac et al. for several years. A nice review of the subject can be found in (Cessac, 2002). Another perspective is based on topological considerations, and has received most of our attention in this thesis. In this perspective, dynamics are characterized mainly on the basis of their trajectories (and hence their attractors and strange or chaotic attractors).

5.2.1 Definition of a dynamical system

The discrete time dynamical systems used in this thesis are usually described with the help of a map $F_{\rho} : \mathbb{R}^N \mapsto \mathbb{R}^N$:

$$x(t+1) = F_{\rho}\{x(t)\}$$

where $x \in \mathbb{R}^N$.

The map $F$ describes how the system evolves in time. It is a function of a family of parameters: $\rho = (\rho_1, \ldots, \rho_p)$.

By iterating the map from an initial condition $x_0$, the system follows a trajectory $\phi(x)$ on the space $\mathbb{R}^N$.

5.2.2 Non-chaotic attractors

An attractor defines a set of points $A$ in the phase space $\mathbb{R}^N$. The set of points remains inside the attractor after iterations of the map $F$:

$$\forall x \in A, \exists y \in A \mid F(x) \mapsto y$$

A basin of attraction $B \supset A$ can be associated with each attractor. This basin represents the set of points in the phase space which tends to converge towards the attractor:

$$\forall x \in B, \exists y \in A \mid \|F^i(x) - y\| \xrightarrow{i \to \infty} 0$$

Similarly, a basin of repulsion can be defined:

$$\forall x \in B, \exists y \in A \mid \|F^{-i}(x) - y\| \xrightarrow{i \to \infty} 0$$

Different types of non chaotic attractors exist:

**A fixed point attractor** defines a set constituted by one point $x = x_\ast$ such that $F(x_\ast) = x_\ast$. The dynamical system behaves statically.

**A periodic limit cycle attractor** defines a set of $n$ points $x_1, \ldots, x_n$ such that $\forall i \in [0, n]$:

$F^n(x_i) = x_i$. When clamped in a limit cycle attractor, the system oscillates periodically among these points.

**A tore or quasi-periodic limit cycle attractor** defines an infinite set of points, the tore results 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 the entire state space $X$. If different attractors co-exist, the border between two basins of attraction is called the separatrix.

### 5.2.3 Definition of deterministic chaos

Chaos is one of these words which most of the time is misused and is associated with some sort of randomness (as in the old world). In our case, deterministic chaos has nothing to do with randomness. It appears in a deterministic dynamical system, and bears a structure which can be qualitatively (and sometimes quantitatively) characterized. Properties of this structure can be very surprising, for example the auto-similarity property which says that when we zoom on a part of the chaotic domain, the same structure is obtained and this infinitely. It is the fractal geometry. Trajectories on these structures are called strange attractors.

Deterministic chaos can appear in dynamical systems which have, according to Feigenbaum, the following characteristics (Feigenbaum, 1983):

- non-linear: only non-linear phenomenon exhibit chaotic behavior;
- recursive: according to Feigenbaum, this complex behavior is mainly generated by its recursive nature.

There are many possible definitions of chaos in a dynamical system. Mathematicians usually use a topological approach (Devaney, 1989).

**Definition 1** $F : V \mapsto V$ is said topologically transitive if for any pair of open sets $U, J \subset V$ there exists $k > 0$ such that $F^k(U) \cap J \neq \emptyset$.

**Definition 2** A subset $U$ of $V$ is dense in $V$ if $\overline{U} = V$

**Definition 3** Let $V$ be a subset of the phase space $X$. $F : V \mapsto V$ is said to be chaotic on $V$ if

1. $F$ has sensitive dependance on initial conditions.
2. $F$ is topologically transitive.
3. Periodic orbits are dense in $V$.

This definition says that a chaotic map possesses three ingredients: unpredictability, non-decomposability and an element of regularity. A chaotic system is unpredictable because of its sensitive dependance on initial conditions. Because of topological transitivity, it cannot be broken down or decomposed into two subsystems (two invariant open subsets) which do not interact under $F$.

However, these concepts of transitivity and denseness are not easy to apply to physical systems. Most of the time, in physics, a dynamical system is said to be chaotic if it presents sensitivity to initial conditions (SIC). This property shows the difficulty of predicting the physical system’s behavior: small errors in experimental readings lead to large scale divergence.
5.2.4 Chaotic attractors or strange attractors

A chaotic attractor, or strange attractor, \( A \) is an attractor which has the following characteristics:

- the distance between two nearby trajectories at time \( t \) in the attractors increases exponentially;
- the dimension of the attractor is fractal.

The dynamical system evolves in an “unpredictable way” in a well-defined region of the phase space. This well-defined region is called the strange attractor.

5.2.5 Stability of fixed points attractors and bifurcation theory

The stability of the fixed point attractors \( x^* \) is computed by considering the behavior of nearby points: a small perturbation \( \eta \) is added to the attractor \( x = x^* + \eta \), and the evolution of this perturbation is analyzed.

\[
F(x^* + \eta) = F(x^*) + \mathcal{D}F(x^*).\eta + o(\eta^2)
\]

where \( \mathcal{D}F \) denotes the Jacobian matrix of partial derivatives of \( F \). The linearized stability problem is obtained by neglecting the terms of order \( \eta^2 \). We have:

\[
F(x^* + \eta) - F(x^*) = \eta(n + 1) \approx \mathcal{D}F(x^*).\eta(n)
\]

The stability analysis of \( x^* \) is reduced to the study of its Jacobian \( \mathcal{D}F(x^*) \), hence to the study of its eigenvalues. The evolution of perturbation in the direction of the eigenvector \( s \) is given by the corresponding eigenvalue \( \lambda_s \):

\[
\eta_s(t + 1) = \lambda_s \eta_s(t)
\]

where \( \eta_s \) is the value of \( \eta \) along the direction of the eigenvector \( s \). Eigenvalues are complex values. Their norm specifies the dilatation: for \( |\lambda_s| < 1 \) the perturbation vanishes, while for \( |\lambda_s| > 1 \), the perturbation increases exponentially in function of the time. The angle of the eigenvalue with the real axis specifies the speed of the rotation.

We can now 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. If \( x^* \) is a periodic point of period \( k \), then \( x^* \) is said to be hyperbolic if \( \mathcal{D}F^k \) does not contain eigenvalues on the unit circle (so its norm is equal to 1).

Depending on the eigenvalues of \( \mathcal{D}F(x^*) \), it appears that three kinds of hyperbolic points exist:

- \( x^* \) is an attractive periodic point if for all the eigenvalues we have: \( |\lambda| < 1 \)
- \( x^* \) is a repulsive periodic point if for all the eigenvalues we have: \( |\lambda| > 1 \)
• $x_*$ is a saddle node in the other cases. This hyperbolic point has at the same time attractive and repulsive behaviors.

A point is not-hyperbolic if at least one of its eigenvalues is on the unit circle. These points are at the origin of the bifurcation phenomenon. This phenomenon is characterized by a qualitative change in the dynamical structure of the system after a modification of a control parameter. Depending on the position of the eigenvalue on the unit circle, there are three generic ways a bifurcation occurs:

$\lambda = -1$: **flip bifurcation**  This corresponds to period-2 oscillation;

$\lambda = 1$: **saddle-node bifurcation**  This corresponds to two branches of stable equilibria;

$\lambda = \|a \pm bi\| = 1, \ b \neq 0$: **Hopf bifurcation**  The dynamics is periodic or quasi-periodic with the angle of rotation $\alpha$ 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 $\pi$, the trajectory crosses a finite amount of points before to cycle repeatedly: we have a periodic regime. If the angle is not a rational fraction of $\pi$, the trajectory never repeats and densely covers the limit cycle. The regime is said to be quasi-periodic. This type of bifurcation is the most usual in complex systems.

Figure 2.5 illustrates these bifurcations on bifurcation diagrams. These diagrams are obtained by the modification of control parameters. The Lyapunov exponent (which is described in Section 5.4.3) is also plotted. This exponent is equal to zero when bifurcations occur and enables the identification of saddle-node bifurcations. This bifurcation does not appear clearly on a bifurcation diagram. For this last bifurcation, only one of the two attractors appears, since for one initial condition the system stabilizes in one attractor. In Section 5.5 these bifurcations will appear more clearly on path-bifurcation diagrams.

5.2.6 Road to Chaos and “types” of Chaos

A road to chaos is the path of bifurcations a system undergoes from a steady state to a chaotic state under impulse of a changed control parameter. When this control parameter changes, the eigenvalues of the Jacobian matrix (2.1) change. A bifurcation will occur when one of these eigenvalues crosses the unit circle. As stated before, the point where it crosses this circle determines the kind of bifurcation.

In order to analyze the roads to chaos, we have to analyze how these eigenvalues are evolving. Different roads exist depending on how eigenvalues cross the unit circle (Albers et al., 1998). In function of the bifurcations undergone, different kinds of chaos are obtained. Three very generic scenarios are described here:

road to chaos by period doubling  Also called the sub-harmonic road, this road to chaos is the most schematic and occurs through a series of flip-bifurcations. Feigenbaum found that the period-doubling cascade of transitions on the road
(a) Flip bifurcation in a one-neuron network with a self-inhibition weight. A neuron network with a self-excitatory positive input acts as control parameter. A negative input acts as control parameter. The bifurcation is indicated by the Lyapunov exponent which is equal to zero.

(b) Saddle node bifurcation in a one-neuron network with a self-excitatory weight. The bifurcation is indicated by the Lyapunov exponent which is equal to zero.

(c) Hopf Bifurcation for a 2-neurons RNN.

Figure 2.5: Bifurcation Diagrams which illustrate the three major kinds of bifurcations. The corresponding Lyapunov exponents curves are plotted below.
to chaos exhibits quite remarkable universal features: there are identical numbers characterizing the ratios of the control parameter for successive period doubling. It does not seem to matter what the details of the system are - which might be rolling flows in a convecting fluid, or oscillations in an electric circuit - there is universality in this kind of transition to chaos (Feigenbaum, 1983).

**Road to chaos by quasi-periodicity** This road occurs through successive Hopf bifurcations. At each bifurcation, a new periodicity is added: we go from a tore $T_1$ (one limit cycle) to a tore $T_2$ (2 superimposed limit cycles), to a tore $T_3$ (3 superimposed limit cycles), etc. The dynamics becomes more and more complex, and at the same time, the system is 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 other. In such cases, we talk about frequency-locked or phase-locked states (Arnold’ tongues). This road to chaos has been described first by Ruelle and Takens (Ruelle and Takens, 1971).

**The intermittency chaos** Point-intermittent chaos is the result of a system at the edge of the saturation. The system is said to be saturated when its parameters are such that the system’s variables are saturated. If the map $F$ of $N$ variables applies $\mathbb{R}^N \mapsto \{-1, 1\}^N$, a variable $x_i$ is saturated if $x_i \mapsto \pm 1$. At saturation, the number of possible states is limited to $2^N$. The system stays on stable attractors. Chaos can appear when the squashing function is not quite saturated. For most of the trajectories the squashing function is saturated and the trajectory tends toward one of the attracting points. Certain combinations of the parameters cause one variable to become unsaturated. A periodic orbit is interrupted before completion by an intermittent point that is not one of the attracting points. After this miss occurs, the system enters a chaotic phase until it goes back to the attractor at saturation of the squashing function. This causes intermittency.

Since this thesis aims at storing information in dynamics of recurrent neural networks, we have obviously dedicated a lot of time to analyzing the different dynamics and more particularly the chaotic dynamics occurring in these networks. From these analyses, we have frequently come across the three above mentioned roads to chaos. However, it seems that the probability to encounter one or another road to chaos is highly dependent on the way the network is built (i.e. its size and its weight matrix). For example, the probability of observing a road to chaos by period doubling shrinks with an increase of the network’s size. The road to chaos by quasi-periodicity appears to be the most natural for random recurrent networks of large size (see also (Dauce, 2000)). This is explained by the fact that the eigenvalues of the Jacobian $\mathcal{DF}$ have larger probabilities to be complex values and hence cause to Hopf bifurcations (Doyon et al., 1993). Intermittency chaos appears to be very generic in large networks, learned through an iterative unsupervised Hebbian procedure (see Chapter 8).

These three roads to chaos are shown respectively in Figures 2.6, 2.7 and 2.8. Each figure shows a representative bifurcation diagram for the corresponding road to chaos.
Moreover, information on the structure of the different chaotic attractors obtained is provided with the help of a power spectrum, a return map and a state diagram for points located inside the chaotic region. It has to be noted that for a chaotic region in the parameter domain, all the chaotic points show more or less the same structure: return maps and power spectra of chaotic points from a same region all look very similar.

These figures emphasize the fact that it is possible to distinguish different kinds of regimes and more particularly different kinds of chaos. Sometimes the expertise can be achieved just by the analysis of one diagram, other diagrams serving only to corroborate it.

However, it has to be mentioned that in practice it is difficult to obtain bifurcation diagrams that show perfect roads to chaos and that are in perfect correspondence with above theory. The number of parameters for a recurrent network composed of \( n \)-neurons are the \( n^2 + n \) weights. To plot a bifurcation diagram we have to make an intersection from a \( n^2 + n \) hyperspace with a one dimensional space (the other dimension of the plot being time). Finding the point of first bifurcation seems to be very difficult. Most of the time, we are “jumping” to bifurcations.

5.3 Theoretical application of the Dynamical System Theory to Recurrent Neural Networks

Recurrent Neural Networks are dynamical systems constituted of interconnected units interacting between each other in a non-linear way. Therefore, in theory, this system should be analytically solvable through the bifurcation theory. However, it appears that the complexity of this study increases drastically with the size \( N \) of the recurrent network.

5.3.1 The one neuron RNN

The system is easily solvable for one neuron with self-interaction (Pasemann, 1993)\(^{12}\). For the evolution equation (2.3) (which is equivalent to the equation used in our model)

\[
x(t + 1) = \theta + w, f(x(t))
\]

where \( \theta \) is the input (or the external stimulus) on the neuron and \( f \) is the activation function, chosen in this experiment as a sigmoidal transfer function.

Three different types of trajectories can occur, revealing three different domains in the \((\theta, w)\)-parameter space (see Figure 2.9). The boundaries of these domains represent the set of values in the parameter space leading to non-hyperbolic points. For these points the norm of the eigenvalue of the Jacobian is equal to one.

For a self-excitatory neuron \((w > 0)\), a hysteresis domain (II) exists over which the system has two coexisting fixed point attractors. This domain is bound by a bifurca-\(^{12}\)It is amazing to note that this study appeared only ten years ago.
5 Dynamical systems theory, chaos and recurrent neural networks

(a) Bifurcation Diagram

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

(c) State versus Time diagram.

Figure 2.6: Period doubling road to Chaos and qualitative structure of this chaos for a 2-neuron RNN. To obtain these graphs, the auto-connection weights have been set to negative values while the weights between them are set positive. Inputs are used as control parameters.
Figure 2.7: Quasi-periodic road to Chaos and qualitative structure of this chaos for a 2-neuron RNN.
5 Dynamical systems theory, chaos and recurrent neural networks

Figure 2.8: Intermittency Chaos and qualitative structure of this chaos for a 3-neuron RNN. Auto-connection weights are highly inhibitive.
tion set, which is determined by a cusp catastrophe at \((\theta_c, w_c) = (-2, 4)\). These units may serve as short-term memory building blocks. For self-inhibitory neurons \((w < 0)\), there is an oscillatory domain (III) corresponding to global period-2 orbit attractors. It starts at the critical point \((\theta_c, w_c) = (2, -4)\). For parameter values outside these two domains, the dynamics only have global fixed point attractors (I).

![Figure 2.9: \((\theta, w)\)-parameter domains. It shows global fixed point attractors (I), bistability (hysteresis)(II), and global period-2 orbit attractors (III)](image)

Figures 2.5(a) and 2.5(b) show bifurcations diagrams for the domains (II) and (III) appearing in Figure 2.9.

It appears that no chaotic orbits exist in this simple system of one artificial neuron network. Chaotic dynamics are observed by adding a linear term of persistence (Pasemann, 1997):

\[
x(t + 1) = \gamma x(t) + \theta + w.f(x(t))
\]  

(2.4)

where \(\gamma\) is a dissipative parameter.

**5.3.2 The two neuron RNN**

The system obtained from a two neuron RNN is also entirely analytically solvable (Pasemann, 1999). It is the simplest neural network having non trivial dynamical properties: for certain parameter domains one finds not only stationary attractors but oscillations of various periodicity, quasi-periodic and chaotic dynamics (without the addition of a persistence state, as in Equation 2.4).

This system is given by a six parameters family of maps \(F_\rho : \mathbb{R}^2 \mapsto \mathbb{R}^2\), \(\rho = (\theta_0, \theta_1, w_{00}, w_{01}, w_{10}, w_{11}) \in \mathbb{R}^6\). The equations giving the boundaries of the multiple domains appearing (fixed points, limit cycle, quasi periodic limit cycle and strange attractors), are not so trivial to solve and even harder to analyze.

Interesting practical results can be obtained for specific parameter configurations, approaching the Hopf bifurcation. For these values the output signal of neurons behave almost sinusoidally. Moreover, the frequency of the oscillators can be controlled by only one parameter (Pasemann et al., 2003).
5.3.3 More than two neuron RNN

No complete theoretical analysis exists for Recurrent Neural Networks having more than two neurons.

This may be for two reasons. The first one being that the equations become very difficult to solve. The second one may be that solutions of these equations would be so complex that they would not help to get a better comprehension of the system, nor to find practical uses for these networks.

5.3.4 Neural Networks of large size

The other perspective is to analyze large connectionist networks by following the road opened more than twenty years ago by Hopfield. He proposed an analogy between neural networks and physics glass spin networks and hence opened neural networks to the well-developed field of statistic physics.

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 (Amari, 1983). The neural network is seen as an emergent system: some global properties, which can be mathematically described, emerge from its individual neurons.

It is also possible to focus on an intermediate level, the mesoscopic level (Cessac, 2002), where the collective behavior of a finite number of neurons is considered.

5.4 Tools used

From a mathematical point of view, the theory of chaos is still in an early stage. However, it provides a paradigm for phrasing situations in mathematics applications. Quoting Holmes:

“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” (Holmes, 1995)

In fact, besides numerical experiments, the sensitivity to initial conditions can be revealed by means of different tools that enable to shed more light on the underlying chaotic process. These tools are presented in this section, dedicating more attention to the ones which will be used in this thesis (see also the famous paper from Eckmann and Ruelle, (Eckmann and Ruelle, 1985)).

It has to be mentioned that most of the tools described here require continuous units. Other tools, working with discrete neurons, exist, however, they have not been investigated in this thesis.

5.4.1 Phase space

The first way to study the dynamical behavior of a system and to detect the presence of attractors is by means of a phase 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. Usually, the axes represent the position and speed of the system, and the corresponding space trajectory or orbit is the evolution of the system.

However, it becomes completely unmanageable while the number of variables in the system increases.

5.4.2 Return map

These maps are obtained by plotting the value of a variable of the system in function of its value at the precedent time step: \( x(n+1) = f(x(n)) \).

The Poincaré map or return map is a very suggestive but hardly conclusive tool for the evidence of chaos and presents a graphical rather than a numerical index of chaos.

5.4.3 Lyapunov Exponent

Lyapunov exponents are convenient indicators of sensitivity to small orbit perturbations, being a characteristic of chaotic attractors. Computations of these exponents are based on the analysis of how two trajectories issued from two infinitesimally nearby initial conditions diverge. This can be interpreted in terms of the Shannon theory of information: two nearby initial conditions, not discernable by some physical conditions will be discernable after a finite amount of time. There is creation of information.

The Lyapunov exponent \( \lambda \) of a one dimensional map gives the average exponential rate of divergence. If we note this initial perturbation \( \eta_s(0) \), the evolution of this perturbation after \( n \) iterations is given by:

\[
\eta_s(n) \sim \eta_s(0) \exp(\lambda n)
\]

The Lyapunov exponent is then defined as (Ott, 1993):

\[
\lambda = \lim_{T \to \infty} \frac{1}{T} \ln \left| \frac{\eta_s(T)}{\eta_s(0)} \right|
\]

By combining the equation (2.5) for the Lyapunov exponent with the equation describing the evolution of the system in function of the eigenvalue (2.2), we have:

\[
\eta_s(n) = \eta_s(0) \lambda_s^n \sim \eta_s(0) \exp(\lambda n)
\]

which gives the relation between the Lyapunov exponent and the eigenvalue of the Jacobian of the system:

\[
\lambda = \ln \|\lambda_s\|
\]

It confirms that non-hyperbolic points (which are responsible for bifurcations) have a zero value Lyapunov exponent.

A \( m \)-dimensional dynamical system, has \( m \) Lyapunov exponents, each of them measuring the divergence rate following one of the directions of the eigenvectors. The
evolution of a hyper-volume $V_0$ is given by:

$$V = V_0 \exp((\lambda_1 + \ldots + \lambda_m)t)$$

The numerical computation of all the Lyapunov exponents is nearly impossible and has not been attempted in this thesis. We calculated the largest Lyapunov exponent by using the following method (Wolf et al., 1984). Analytical computations of Lyapunov exponents through analyses of the Jacobian matrix is reserved for future works.

First, before starting the calculation, we simulate the network during a sufficient amount of time steps, to help ensure that all transient effects are avoided (so that if an attractor exists, we would be sufficiently close to it).

We begin the computation of the Lyapunov exponent at that time, let’s say $t = 0$. From the state $\vec{x}(0)$ of the network (where $\vec{x}(t)$ defines the vector state of all neurons at time $t$), a nearby state $\vec{x}'(0)$ is computed with a separation $\eta$.

We define $\Delta y_t$ the vector between these states:

$$\Delta y_t = (x_1(t) - x'_1(t), \ldots, x_n(t) - x'_n(t)) \tag{2.8}$$

So by construction, we have the distance $|\Delta y_0| = \eta$. The network is simulated with each of these states. We obtain $\vec{x}(1)$ and $\vec{x}'(1)$. The distance between the two states is computed and the ratio

$$\frac{|\Delta y_1|}{|\Delta y_0|} \tag{2.9}$$

is recorded. The vector $\Delta y_1$ is then rescaled to a length $\eta^{13}$. The new state $\vec{x}'(1)$ is computed following:

$$\vec{x}'(1) = \vec{x}(1) + \Delta y_1 \tag{2.10}$$

This process is repeated, and the largest Lyapunov exponent is estimated from the average of the logarithm of the scalings:

$$\lambda = \frac{1}{T} \sum_{i=0}^{T-1} \ln \frac{|\Delta y_{i+1}|}{|\Delta y_i|} \tag{2.11}$$

where $T$ is the number of iterations of the network from which the average is taken.

Below are the expectations for Lyapunov exponents:

- $\lambda < 0$ for fixed or periodic points. We are on a stable attractor. If we modify the state slightly, it will converge again and the modification will vanish.

- $\lambda = 0$ for quasi-periodic points. We are on a limit cycle, after a perturbation, the system will continue on a nearby road.

- $\lambda > 0$ for chaotic points. The system is highly sensitive to initial conditions. Any perturbation will cause the network to diverge.

\[13\]This rescaling is a computational trick which has proven to be more robust on numerical computations (Wolf et al., 1984)
A measure of the “chaoticity” of the network is computed by averaging Lyapunov exponents obtained from a huge set of different inputs, with different initial conditions.

To compute other Lyapunov exponents, we have to create an initial perturbation which has no components along the direction of the eigenvector having the biggest eigenvalue. This is done to avoid that this component totally shadows the other ones. It implies that we have to determine the direction of the eigenvectors. This has not been done here.

5.4.4 Spectral Analysis

A well known numerical tool is the spectral analysis allowing the analysis of dynamical systems’ periodicity. The power spectrum of a signal is defined as the square of its Fourier amplitude per unit time (Eckmann and Ruelle, 1985). Typically it measures the amount of energy per unit time contained in the signal as a function of the frequency.

The power spectrum allows us to distinguish periodic, quasi-periodic and chaotic series. Moreover, it enables the differentiation between the different kinds of chaos.

- The power spectrum is continuous and equal to zero for a fixed point attractor;
- The power spectrum has one peak for periodic outputs (cycles of period-n), and subsequent peaks for its harmonic;
- For quasi-periodic outputs, the power spectrum shows important peaks on the frequencies of the periods, and some smaller peaks for the harmonic frequencies of the main frequencies;
- The power spectrum of chaotic outputs, is something between the power spectrum of quasi-periodic outputs and the power spectrum of a random output. In the random case, the power spectrum reflects broad band noise.

However, this distinction calls for cautiousness. It may very well be that an output seems to be chaotic or random without it actually being so. This may be caused by the fact that the length of the observation interval is smaller than the periodicity of the system.

An interesting issue is the global topology of the strange attractor that results from the chaotic RNN, and the power spectrum of the associated dynamics. Figures 2.6, 2.7 and 2.8 show the power spectrum associated with different kinds of chaos. They are clearly distinguishable. For example, for a chaos obtained from a quasi periodic road, power spectra typically have multiple dominant incommensurate peaks and an approximatively white background three or four orders of magnitude below that of the dominant frequency. Therefore the global topology of the attractor resembles a limit cycle or higher dimensional torus, perturbed by chaotic deviations.
5.4.5 The Hausdorff dimension

This quantity has been brought very much to the attention of physicists by Mandelbrot, who uses the term fractal dimension. Ever since, it has also been used as a generic name for different mathematical definitions of dimension for “fractal” sets.

Let $A$ be a compact metric space which defines the attractor analyzed, and $N(r, A)$, the minimum number of open balls of radius $r$ needed to cover $A$. The dimension of the attractor is defined by:

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

The computation of the Hausdorff dimension has been implemented in the platform developed during this thesis. However, since the graphs obtained appeared redundant due to other results, no plots will be presented.

5.5 Initial conditions, basins of attractions and hysteresis

These three notions are capital in our problematic and more generally in any pragmatic investigation of dynamical systems.

**initial condition** It defines a point, in the $N$-dimensional space, where the network lies at initialization. For recurrent networks, it can be related to the initial history of the system.

**basins of attraction** As defined above, it represents the set of points in the phase space (here $\mathbb{R}^N$) which tends to converge to an attractor.

For the one neuron RNN, we have seen that for some parameter configurations, two attractors (hence two basins of attractions) coexist. The network’s dynamics converges to one of the two attractors regarding its initial condition. For higher dimensional RNN (and larger parameters domains), various dynamics may 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** This phenomenon may appear when the recurrent network is evolved through control parameters (which can be the weight of a connection, the external stimulus, etc.), and is due to the coexistence of multiple basins of attractions: when a control parameter is changed, from one value to another one and then in the reverse direction, different initial conditions are crossed going in one way or in the other one. As a consequence, the forward pass and the backward pass result in different bifurcation diagrams.

Figure 2.10 shows the hysteresis phenomenon occurring in a one-neuron RNN while the input $\theta$ is changed. This hysteresis results from the existence of a domain of
Figure 2.10: Path Bifurcation for one-neuron RNN. A simple hysteresis appears to be corresponding with a bistable domain (see section 5.3). The Lyapunov exponent is equal to zero when saddle-node bifurcations occur.

Figure 2.11: Path bifurcation diagram for a 3-neurons RNN. It shows a complex hysteresis phenomenon. Depending on the way the control parameter is changed, different routes are followed. It appears that for large parameter domains, depending of the internal state, the system evolves in different attractors.
bi-stability 5.3. Figure 2.11 shows a more complex hysteresis phenomena appearing in a larger RNN.

So different initial conditions 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. After this bifurcation, dynamics are often not global.

This hysteresis phenomenon appearing in one-neuron RNN has suggested the use of these “networks” as short-term memory building blocks (Pasemann, 1997). It has also been demonstrated that hysteresis effects can improve performance for robots trained to support navigation tasks (Huelse and Pasemann, 2002).

6 Conclusion

The aim of this chapter was to familiarize the reader with the multiple notions underlying our research. First, basic notions of neurophysiology were introduced and some more attention was given to neuro-dynamical observations.

Following, a description was given to their formal counterpart, the large field of connectionist networks. 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.

This was followed by an introduction on nonlinear dynamical systems theory – more specifically the theories of bifurcation and chaos. In the field of neuroscience, this theory has provided powerful new concepts for analyzing and interpreting data. Following Skarda and Freeman ((Skarda and Freeman, 1990a) and (Skarda and Freeman, 1990b)), this forces a revolution in the practice of neuroscience: “In our research, the realization that self–organized and chaotic dynamics are essential to brain function has led us to reject the underlying explanatory framework that made reductionism the hallmark of scientific explanation.”.

In the field of artificial neural networks, using the dynamical systems theory has allowed the entire solving of the one neuron RNN, and has given some predictable results for the two neuron RNN. When the network’s size is increased, because of the high-dimensional parameter space, theoretical analyses become impossible. Two options remain when facing large networks: to use statistics mechanics or, as adopted in this thesis, to analyze the topology of the attractors from a practical point of view.
Chapter 3

State of the art and Methodology

1 Introduction

After the summary of the preliminary notions presented in Chapter 2, we can now present an outline of the methodology developed in this thesis. Roughly speaking, the goal of this thesis is to try and find a way to effectively store information in the dynamics of artificial neural networks, in a similar way the brain does. Moreover, in this thesis there is the underlying belief that qualitative and quantitative analyses of the different chaotic dynamics found in these artificial models will help us reach our goal.

Different groups of researchers share the idea that connectionist networks will find convincing applications when used in a dynamical way. Moreover, they share the belief that such research could be guided by recent neurophysiological and dynamical observations of the brain. The approach followed by three of these groups will be briefly discussed in Section 2. Our approach is discussed in Section 3.

2 Related approaches

2.1 Freeman and Kozma et al.

Freeman from the neurophysiological laboratory of Berkeley has conducted seeding neurophysiological experiments for 20 years, showing evidence of complex activity occurring in the brain through observations of EEG signals obtained from different brain locations (Skarda and Freeman, 1987).

Together with Kozma from the computer science laboratory of Memphis University, he adopted the approach to build artificial models replicating at best brain dynamics (Kozma and Freeman, 2001). For this purpose, complex architectures built on simpler ones were developed:

“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 spatiotemporal neural developments at the mesoscopic and macroscopic scale” (Muthu et al., 2004)

For example, the KIV-model, the last one, consists of four components: the hippocampus (for the spatial orientation - modeled with a KIII set), the cortical region (for sensory inputs - KIII set), the midline forebrain (deals with the internal goals - KIII set), and the amygdala.

This approach has two goals: to learn more about neural assemblies and to develop autonomous robots capable of performing cognitive tasks.

### 2.2 Doyon and Samuelides et al.

This group has been created in 1990 under the impulse of Doyon, neurophysiologist at INSERM, and Samuelides, mathematician at Supaéro.

From the beginning they decided to analyse large fully connected recurrent neural networks. This enabled them to compare dynamical behaviors of these models with the mathematical results obtained at the thermodynamic limit (when the network’s size tends to infinity) (Doyon et al., 1993), (Doyon et al., 1994). This mathematical approach has its roots in the work done by Amari (Amari, 1983) and consists of describing the model formed by a huge number of microscopic variables using some macroscopic observables. In fact, deep mathematical analyses of the model of two populations of neurons (one population of inhibitory and one population of excitatory neurons) –which is the KII model of Freeman– have been performed.

More recently, a lot of attention has been given to using the obtained knowledge in autonomous robots. However, while Freeman et al. tried to obtain cognitive behaviors by “creating an artificial brain”, here they tried to obtain the same cognitive behaviors by finding Hebbian based learning procedures (Dauce et al., 1998).

### 2.3 Pasemann et al.

A very recent neuro-dynamics approach has been initiated by Pasemann from the Fraunhofer Institute for Autonomous Intelligent Systems.

Theoretical results on the one neuron (Pasemann, 1993) and the two neuron network (Pasemann, 1999) were obtained and applications were proposed based on these results. Examples are the development of one chaotic neuron (Pasemann, 1997), or the creation of a generator of frequencies using a two neurons network (Pasemann et al., 2003). All these results have been qualitatively compared with brain dynamics (Pasemann, 2002).

Pasemann followed the same philosophy as the two groups already mentioned by trying to use this kind of networks in autonomous robots. However, the question of the architecture –what type of recurrent structure is to be used for the generation of a successful behavior– was tackled in a different way. While Freeman et al. created a complex architecture (the KIV model) and Doyon et al. used a simple model of huge
size, Pasemann developed an evolutionary algorithm for the structural development of neural networks that uses a fitness function (Huelse and Pasemann, 2002). This algorithm is called ENS (for evolution of neural systems by stochastic synthesis).

An interesting result observed by Pasemann is that most often well skilled robot behaviors are partly due to hysteresis effects (cf. Section 2.5.5) associated to specific recurrences (Huelse and Pasemann, 2002).

3 Our approach

Like the other groups, we are anchored to the dynamical hypothesis (Skarda and Freeman, 1990a), (Guillot and Dauce, 2002)) roughly saying that information in the brain is principally realized by the dynamical relation among neuronal states, with the viewpoint that a neural correlate of cognitive behavior exists. By comparing artificial neural models with biological models, a working analogy between the brain exhibiting naturally chaotic behaviors and small size dynamical connectionist models is searched for.

However, when compared to the other research groups, a major distinction arises in how they use their connectionist models. According to neurophysiological observations the importance of the environment appears to be crucial in learning: 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 (this approach has its roots in the animat approach developed by Brooks in 1986 (Brooks, 1986)). Following this idea, the three other groups use connectionist models in autonomous robots, and analyzed the impact of learning mechanisms in the network’s underlying dynamics in conjunction with the observed behavior. The idea is to see how the robot controls “himself” by using an artificial brain which processes information from the environment and from past history to build a new internal representation.

Such learning task has one advantage and one disadvantage. The advantage is the focus on a very basic learning mechanism which does not seem to rely on high cognitive processes: the processus of adaptation in an environment. The disadvantage is its reliance on robots, sensor technology, etc. demanding a lot of expertise, and a lot of technical and financial resources. In order not to depend on technological difficulties, the methodology we developed is quite different:

Following the idea that information is stored in the brain in almost cyclic dynamics while attentive waiting states correspond to chaotic dynamics, we argue in this thesis that artificial RNNs could be used to store information in the different attractors of the occurring dynamics.

At this level, the type of information stored does not need to be specified, it could be high-level data stored in memories, but also (and more probably) it could be information driven by a robot controller.
From this point of view, we do not claim a local biological precision, neither at the level of the simple neuron, nor at the level of their dynamics. Instead, we concentrate ourselves on simplifying the biological complexity in order to identify the simple control variables of the system. To achieve this we started with an abstract model of neurons and investigated the different dynamics occurring in such a network.

However, we do share Freeman’s idea that a cognitive process will be obtained through the development of complex architectures made of simpler components (the KIV is made of KIII which in turn is made of KII, etc.). In our work, this has led to the development of a neural platform which is perfectly suited for the future development of hybrid models and complex neuronal architecture.

3.1 What “information” means in this thesis

As stated in the title, the goal of this thesis is to store information through complex dynamics in recurrent neural networks. Using neural networks to manipulate information is not a new idea. In fact, most of the research conducted in this field is supported by the view that what the brain does is processing information.

The brain processes information in many ways. First of all, information received from external stimuli is globalized. The data is analyzed in the light of some personal history, interpreted and associated with other known data. An action plan is decided on and information is sent to its effectors (muscles) while controlling and correcting their actions.

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 first big family of neural networks, the feedforward neural networks, has been used principally to associate information in a hetero-associative way: a given input is associated with a given output having a different syntax and/or semantic (the input and the output are different pieces of information). Feedforward connections are such that the generated output is fully determined by the given input: a given stimulus gives rise to a determined action. This type of networks has been widely used for clustering and classification, among others (Haykin, 1998).

By adding some internal recurrences, the output is no longer uniquely determined by the input: the generated action (the output) is a function of the given stimulus and of the network’s internal states. This principle has given rise to the wide family of feedforward recurrent networks composed of many different kinds of architectures (Kremer, 2001). Typical applications for these networks is the simulation of context-sensitive dependencies required for language processing.

In 1982, Hopfield proposed a new way of using connectionist networks. To reach his goal, he developed a new kind of architecture, the fully recurrent neural network,

---

1This is confirmed by the name of one of the most prestigious conferences held in the domain: the NIPS (“Neural Information Processing Systems”) conference
nowadays known as Hopfield network. Here, inputs are no longer seen as external, they directly initialize the internal states of the neurons. It is no longer an “external” stimulation but rather an internal modification that forces the network into an initial location of its phase 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. 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 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). This defines memories which are addressable by the content in contrast with computer memories, which are retrieved by an address (and hence any noise freezes any recognition).

This thesis follows Hopfield’s idea of storing information in the network’s dynamics with some content addressability. However, we added limit cycles attractors to the already used fixed points attractors. These limit cycle attractors appear naturally once the constraint on the weight symmetry is relaxed. Another crucial difference (which will prove a greater performance) is the usage of the input data not only to initialize the internal states, but also as an external stimulus. The idea is that this external stimulus will modify the inner dynamics, hence, increasing the network’s dynamical potential.

The information is the association between the stimulus and a limit cycle attractor. After learning, if the network receives a specific learned stimulus (or a slightly noisy version of this stimulus), and if its initial internal state is not too far from the expected limit cycle attractor (at least, from one of its patterns), the network’s dynamics is expected to converge to this limit cycle attractor.

The notion of external stimulus will keep the same meaning all along this thesis.
This is not the case for the notion of limit cycle attractor. Therefore each chapter will have to define precisely what is meant by “learned cycles”, the reason being that the formulation of a coding scheme at the level of the single neuron is not the main concern of this thesis. We rather base our descriptions on “observables” that represent macroscopic behavior observed as a collective motion. For this purpose, in Chapter 4 and 5, the network’s dynamics will be observed through the mean signal of the network. However, learning in global limit cycles attractors will no longer be possible in Chapter 6 and 7, while relying on local Hebbian algorithms. Hence, in these chapters, learned cycles will have to be fully specified, both spatially and temporally. In Chapter 8, an unsupervised version of the local Hebbian algorithm will be introduced to avoid the reliance on local neural descriptions. The results in term of encoding capacities will be impressive.

3.2 Description of the model

Recurrent systems are inherently dynamical. In contrast with static feedforward networks, where the neural behavior can be seen as a passive reaction fully determined by a given stimulus, recurrent networks lead to self-organized processes where response to a stimulus can be seen as an active process.

As mentioned in a previous chapter, these networks have been used in different areas. Nearly each of these uses has led to a specific formal model. However, none of them has given convincing enough results leading to practical applications, and none has emerged as a generic model usable in many different contexts.

Since the aim of this thesis is to store information in these networks, we did not have any specific architectural requirements. Following the steps of Hopfield, we decided to use a fully connected recurrent neural network.

Each neuron’s activation is a function of other neurons’ impact and of an external stimulus. The mathematical description of such a network is the very classical one. The activation value of a neuron \( x_i \) at a discrete time step \( n \) is:

\[
\begin{align*}
    x_i(n+1) &= f(\text{net}_i(n)) \\
    \text{net}_i(n) &= \sum_{j=1}^{N} w_{ij} x_j(n) + w_{i\iota} \iota_i(n)
\end{align*}
\]

(3.1)

with

- \( N \): the number of neurons of the network;
- \( f \): the saturating activation function;
- \( w_{ij} \): the weight between the neurons \( j \) and \( i \);
- \( w_{i\iota} \): the weight between the external stimulus and the neuron \( i \);
- \( \iota_i \): the external stimulus for the neuron \( i \) at time \( n \);

In this thesis we have chosen to call \( \iota \) the external stimulus (or stimulus) of the network. However, in the literature (and in previous papers), the same term can be found under different names, for example: the input of
the network or its bias. Even though they all have the same mathematical meaning, these words have a different connotation. The feedforward (and recurrent-feedforward) approach use the term “input”: the network is seen as a passive system generating an output while receiving an input. In Hopfield’s model, the term “bias” is used to describe a global and immutable perturbation of the network. The name external stimulus has been explicitly chosen to suggest neurobiological parallels and to reinforce the idea of a self-organizing system.

As can be seen, this model works in discrete time steps, in contrast with time continuous models which are based on differential equations used in theoretical analysis. This choice has been made to ease numerical simulations.

Three important choices remained: the choice of the activation function, the update rule and the weight configuration studied.

With regard to the activation function, two opposed types will be used in this thesis: continuous and discrete activation functions. However, in both cases the function will be saturating (i.e. bounded in positive and negative values).

- Two types of continuous activation functions have been used, the classical tanh function and the sigmoid function. Their main difference lies in the range of values obtained for the neuron’s state. (A third activation function, described below, has been introduced for practical reasons).

\[
\begin{align*}
  f(x) &= \tanh(gx) & f : \mathbb{N} \rightarrow [-1, 1[ \\
  f(x) &= \frac{1}{2}(1 + \tanh(gx)) & f : \mathbb{N} \rightarrow [0, 1[ 
\end{align*}
\] (3.2)

where \( g \) is the slope parameter;

- For the discrete case, the only activation used is the classic sign function which allows to work with bit-patterns.

\[
  f(x) = \text{sign}(x) = \begin{cases} 
    1 & \text{if } x \geq 0 \\
    -1 & \text{if } x < 0 
  \end{cases}
\] (3.3)

The choice between synchronous and asynchronous updating appears critical for the underlying dynamics (Haddou, 2003). For example, the proof that recurrent networks converge to stable fixed point attractors relies on an asynchronous update rule (Hopfield, 1982).

Weights are the system’s parameters. In this thesis, different weight configurations will be analyzed. In the first part, uniform random configurations are studied, then weight configurations obtained through different learning algorithms are tested.

Unless otherwise mentioned, the dynamics will be characterized, most of the time, with the help of the observable \( m_N(t) \) defined as the mean signal of \( x(t) \) (as explained, in this thesis we are concerned rather with macroscopic behavior than with formulat-
ing a coding scheme at the level of the single neuron):

\[ m_N(t) = \frac{1}{N} \sum_{i=1}^{N} x_i(t) \] (3.4)

Practically, this mean signal is computed by adding a meaner neuron, i.e. a neuron having a meaner activation function. This model is represented in Figure 3.2.

Figure 3.2: Fully connected recurrent neural network of 3 neurons using a sigmoid activation function. All the neurons are connected with a fourth neuron using a meaner activation function. This fourth neuron enables the computation of the mean activation signal without losing a full neuronal model.

3.3 The methodology used: development of a neural toolbox

There has been no straightforward direction to reach the goal of this thesis, i.e. to store information through complex dynamics in recurrent neural networks. Moreover, at the beginning of our work it was not clear how to understand this learning task. At that time, the main idea was to analyze the dynamics of RNNs with the belief that it would help in finding engineering and/or cognitivist applications.

The methodology applied could be stated as follow: focus our analyze not only on direct results of networks’ encoding capacities, but also on dynamical observations of these networks before and after learning procedures. This methodology has proven good results and has led us to the development of an unsupervised Hebbian learning algorithm. This algorithm enables the storage of huge amounts of information in RNNs without letting chaotic dynamics spreads out all over.

Since it has been proven in Chapter 2 that theoretical analyses are very cumbersome, even for a system constituted of only two neurons, we decided to go for a more practical and empirical way. After reviewing the existing neural softwares, it appeared that no platform exists that is generic enough to provide different kinds of dynamical tools for any kind of neural architecture. Therefore, we decided to build our own tool, named the “Neural Development Kit” after the second release.

This neural platform is currently at its third release and contains more than two hundred and fifty classes and more than sixty five thousands lines of code. It enables the graphical construction of any imaginable neural architecture and to observe the
underlaying dynamics of the network obtained using several dynamical tools. Each neuron can have its own parameters and its own transfer function. Moreover different learning mechanisms have been implemented, most of them being original contributions from the author. The current version of this neural platform is briefly described in Annex A.
Chapter 4

Dynamical observations of random networks

1 Introduction

The goal of this work is to try to encode data on cyclic regimes of recurrent neural networks (RNN). This idea comes both from neurophysiological observations already described in previous chapters, and from dynamical observations of random recurrent neural networks of small size where this kind of potential appeared. This chapter aims at describing these observations.

The connectionist model used is a fully connected recurrent neural network, working in discrete time steps and with continuous state neurons (see Section 3.2). The activation of a neuron at a given time step is a non-linear function of the sum of all the neuron’s activity pondered by the weight of the connections and of an external stimulus. For a given stimulus, RNN dynamics are highly dependent on the network’s initial state (see Section 2.5.5). In this chapter and in the following one, the stimulus feeding the network will also initialize the network’s initial state.

2 Storing information in the dynamical attractors

2.1 An infinite potential

When Hopfield decided to encode information in RNNs, the system was fully determined by the initial condition on the internal states (Hopfield, 1982). Moreover, by constraining the network to follow an Hebbian learning procedure the network was defined as a dissipative system and hence dynamics were restricted to converge to fixed point attractors.

The model used here has two major differences with the Hopfield model. First of all, there are no similar Hebbian prescriptions constraining the network to stable dynamics. The second difference is that the system is determined not only by the initial internal states, but also by the external stimulus. As a consequence, the network
is no longer attracted to a limited set of fixed point attractors, but instead the
network has an infinite number of attractors having variable dimensions. The impact of
the stimulus is the following: for the same initial condition, each different stimulus
modifies the inner dynamics of the model, and leads the network, through a series
of bifurcations, to a different attractor. These attractors can be qualitatively different,
for example going from a period-2 cycle to a period-4 cycle or going to a stable fixed
point to a chaotic attractor, or can differ simply quantitatively, i.e. only differences in
amplitudes appear.

Figure 4.1 shows the evolution of the mean signal $m_N(t)$ when the system is sub-
mitted to 4 different static stimuli $I_1, I_2, I_3, I_4$, for a period of 50 time steps each time.
The four stimuli are randomly extracted from a given neighborhood. The two first
stimuli lead to similar but quantitatively different activation patterns. The others
show very different dynamical regimes, both in periodicity and repartition of activa-
tion. This illustrates the sensitivity of our model to small stimuli changes.

![Figure 4.1](image)

Figure 4.1: Evolution of the dynamics while the system is successively sub-
mitted to 4 external stimuli randomly extracted from a given neighborhood,
$I_1, I_2, I_3, I_4$. The mean signal $m_N(t)$ and its return map are represented.

We define the following sets of functions and periodic functions:

$$T_i \equiv \{ f \mid f(x) = f(x + ki) \quad \forall k, x \in \mathbb{N} \}$$
$$T_\infty \equiv \{ f \mid \exists i \in \mathbb{N}, f(x) = f(x + ki) \quad \forall k, x \in \mathbb{N} \}$$
$$T^*_i \equiv T_1 \cup T_2 \cup \ldots \cup T_i$$
$$T^* \equiv T_1 \cup T_2 \cup \ldots \cup T_i \cup \ldots \cup T_\infty$$

Thus, $T_i$ represents the set of functions of period $i$, $T_\infty$ represents the set of non-
periodic functions, $T^*_i$ represents the set of functions of period lesser of equal to $i$
and $T^*$ represents all the functions.

Provided with these definitions, the mapping between the input space and the
output space can be specified. The external stimuli are constant values through time:
$\iota_i \in T_1 \quad \forall i$, and belong in the hyperspace $\mathbb{R}^n$. The network’s mean signal can have
different dynamical behaviors: $m_N \in T^*$, and due to the activation function, it is
restricted to values in $[0, 1[$:
The large potential of this model comes from the use of continuous neurons in spite of discrete neurons. However, this richness also comes with a pitfall: even an infinitesimally slight modification of the stimuli can lead to a modification of the inner dynamics and thus of the attractor. No robustness can be assumed: to recognize a given output, we have to work with a high precision number. Any noise can freeze the recognition.

2.2 Model Quantization

In order to be able to say that two close output values lay in a same equivalence class, the output space is divided into \( r \) intervals of equal size, yielding to \( r \) partitions in the output space (each partition is one equivalence class). As a result, the temporal series of real value on the output is seen as a symbolic series. This kind of symbolic dynamics becomes more and more popular to extract invariant grammatical and statistical characteristics from time series (Ebeling and Nicolis, 1991), (Hao, 1991). As proposed in (Omlin, 2001), we refer to \( r \) as the quantization level. Such quantization is obtained naturally by adding an output filter layer. The set of these equivalence classes is defined as:

\[
\mathcal{A}_r = \{a_1, a_2, \ldots, a_r\}
\]

The resulting dynamics generate strings, composed of members of \( \mathcal{A}_r \), in the following way: at each time step, the letter to add is the one associated with the interval containing the mean signal \( m_N(t) \). Thus we have:

\[
m_N(t = k) \in ]0, 1[ \mapsto \mathcal{A}_r \quad k \in \mathbb{N}
\]

Each external stimulus of the learning set triggers and thus becomes associated with a fixed point or a cyclic output. The maximum period of such cyclic output (noted \( p \)) is a further parameter of the simulation. So we have:

\[
m_N(\cdot), x_i(\cdot) \in T_p^*
\]

and the mapping in equation (4.1) becomes:

\[
\begin{align*}
\mathbb{R}^N & \mapsto ]0, 1[ \\
T_i^N & \mapsto T_p^*
\end{align*}
\]

For instance, tuning \( p \) to 1 forces the network to only deliver symbolic fixed points in response to any external stimulus.

Finally, in order to guarantee the robustness of the learning, in a way reminiscent of the original use of Hopfield Nets as associative memory, the same dynamical output must be associated with a variety of external stimuli, all selected in a small
Dynamical observations of random networks

A hypersphere of diameter $\epsilon_{rob}$ around the original external stimulus to learn. Such an external stimulus $\bar{\iota}$ is say to be $\epsilon$-robust iff $\forall \bar{\iota}' \mid \mid \bar{\iota}' - \bar{\iota} \mid \leq \epsilon$ we have the same quantized output. $\mid \mid \bar{\iota}' - \bar{\iota} \mid \mid$ is the euclidian distance between $\bar{\iota}$ and $\bar{\iota}'$.

Thus the network behaves as a robust hetero-associative memory.

2.3 Example

Below you will find an example of a practical external stimulus-output data for this model. We set the quantization level $r$ to 4 and the maximum period length $p$ to 4. Since $A_r$ must contain $r$ different symbols for representing the different class of equivalence, we can define:

$$A_4 = \{a, b, c, d\}$$

with $[0, \frac{1}{4}] \mapsto a$, $[\frac{1}{4}, \frac{1}{2}] \mapsto b$, $[\frac{1}{2}, \frac{3}{4}] \mapsto c$ and $[\frac{3}{4}, 1] \mapsto c$.

We define the set $\delta$ as the set of all length strings less or equal to $p$ without redundancy:

$$\delta = \{a, b, c, d, ab, ac, ad, \ldots, ddda, ddde, ddcc\}$$

since $dddd \equiv dd \equiv d$, only $d$ appears in $\delta$.

The set of all the potential cyclic attractors of the network, $\Theta$, will be a subset of this set: $\Theta \subseteq \delta$. This set is dependent of the network’s configuration.

Let us now continue our example with a 3-neurons RNN. In this case the input set $I$ is composed of a set of external stimulus $\bar{\iota} = \{\iota_1, \iota_2, \iota_3\}$, where $\iota_i$ is the input value for neuron $i$.

The network maps $I \mapsto \Theta$. For this example, the mapping defined in 4.5 becomes:

$$\begin{align*}
\Re_3 &\mapsto A_4 \\
T_1^N &\mapsto T_4^s
\end{align*}$$

(4.6)

The first term states that the external stimulus is in a 3-dimensional space and will be transformed in a one-dimensional output represented by the mean signal. The second term of the mapping shows that the 3-dimensional stimulus is constant through time while the mean signal in the output is of period less or equal to 4. Figure 4.2 shows such a mapping for a 3-neurons RNN. In this example, two nearby external stimuli are mapped to the same symbolic string after quantization of the mean signal.

Coming back to Figure 4.1, this explains how two qualitatively similar and quantitatively nearly equal attractors obtained from the nearby inputs $(\bar{\iota})_1$ and $(\bar{\iota})_2$ lead to different quantized outputs, here $adc$ and $adb$. However, at another quantization level, they could become equal.

2.4 Model’s Capacity

The encoding potential of the RNN is the quantity of information which could be stored in symbolic strings obtained from the quantization of the network’s mean sig-
Figure 4.2: Two nearby constant inputs for a 3-neurons RNN are mapped to cyclic outputs. The two outputs are slightly different, this difference disappears after quantization. Another input maps to a different period 2 outputs.

2 Storing information in the dynamical attractors

This amount of information is stored in a dataset:

\[ D = \{ D^1, \ldots, D^q \} \]  

(4.7)

The quantity of data stored in the network is quantified by defining a load parameter \( \alpha = q/N \). Each data \( D^\mu \) is defined by a pair composed of:

- a pattern \( \chi^\mu \) corresponding to the external stimulus feeding the network;
- a string \( S^\mu(p, r) \) of size lesser or equal to \( p \), and composed of letters belonging to the alphabet \( A_r \).

2.4.1 Model’s theoretical maximum capacity

Because the strings obtained from the quantization of limit cycle attractors of the dynamics are limited, the model has a theoretical maximum encoding capacity. This capacity is a function of the quantization level \( r \) and the maximum string size \( p \). The computation of the total number of strings appears to be cumbersome. However, a sufficient valuable estimation can be obtained from the following simplistic evaluation: since a string of size \( p \) is composed of \( p \) characters, each of them having their value taken from an alphabet of size \( r \), the theoretical capacity is given by:

\[ q_{M,\text{theor}} = r^p + o \]

where \( o \) takes into account the following considerations.

- this capacity is overestimated for the two following reasons:
  - permutations are redundant. For example, \( abc, bca \) and \( cab \) correspond to the same limit cycle attractor;
  - if a period appears inside a string, this string is in fact equivalent to the string given by the inner period. For example, \( bbb, bb \) and \( b \) correspond to the same fixed point attractor;
• this capacity is underestimated because strings of size smaller than \( p \) are also accepted. Hence, terms in \( r^{p-1} \) appear.

For example, if the quantization level \( r = 8 \) and the maximum size of the limit cycle attractors is \( p = 4 \), the maximum theoretical capacity is given by \( q_{M, \text{theor}} \approx 4096 \). This theoretical limit does not seem constraining at all. However, if the quantization level is set to \( r = 2 \) (which means dealing with binary strings), this maximum theoretical capacity shrinks. Because in Chapters 6, 7 and 8, the quantization level will be set to \( r = 2 \) to enable the comparison with images made of bit-patterns, a modification of the model will be required.

# 3 Numerical results of encoding capacity

Now that the model is fully described, and that we have defined and explained what we mean by the storing capacity of the network, we can focus on practical observations of the networks.

In the first part of this section, encoding capacities in terms of the number of robust attractors are computed. For each network, the capacity obtained is related to its mean Lyapunov exponent. The mean Lyapunov exponent is computed in the following way: a huge set of random inputs (or stimulus) is presented to the network. For each associated resulting dynamics, the biggest Lyapunov exponent is computed. The result is the average of these values. The computation of the Lyapunov exponent is done following the method described in (Wolf et al., 1984) (see also Section 2.5.4.3). The second part will give a first glance at qualitative observations through return maps and power spectra.

## 3.1 Quantitative Results

Statistical results are provided in this section: a huge number of networks having different matrix of connectivity are being compared. For each network studied, the encoding capacity is computed by testing the resulting dynamics for 5000 different stimulus. For each stimulus \( \iota \), if the corresponding quantized output is a robust cycle of period lower or equal to \( p \) and has not been already obtained previously, it is added to the number of potential attractors. The total number of potential attractors gives the encoding capacity. The quantization level \( r \) is set to 8 and the maximum period \( p = 4 \).

### 3.1.1 The main result

Different results will be compared in this section: networks of variable size, networks having different weight distributions, networks with neurons having variable activation function, etc.

Each time, specific results will be discussed. However, when the “complexity” of the network, represented by the mean Lyapunov exponent, is plotted in function of
the number of potential attractors, all plots obtained have the same shape. This shape appears in Figure 4.3.

![Figure 4.3: Typical plot showing the complexity of 200 random networks in function of their encoding potential. Each point represents a different random network. It appears that networks having a huge encoding potential (region (a)) are more “complex” and that stable networks (region (b)) have a small number of potential attractors.](image)

Typical values for the number of attractors are in the range of hundreds. This calls our attention to two findings. The first one is that using cyclic regimes to store information can boost the encoding capacity compared to the standard Hopfield Network where the total number of attractors was equal to 14% of the network’s size. Of course, the comparison is not really fair: Hopfield provided a learning algorithm for auto-associative memories, while we are studying the encoding potential in a hetero-associative way. However, both cases are dealing with attractors existing in fully connected RNN.

The second finding, and maybe the main result, is the correlation appearing between the number of potential attractors and the index of the “chaoticity” of the network given here by the mean Lyapunov exponent: networks having a huge encoding potential (region (a)) are more “complex” and stable networks, i.e. networks having low Lyapunov exponents (region (b)), have a small number of potential attractors. Apart from that, no useful information is received: for example, if one network has less potential dynamical attractors than another one, we cannot conclude that this network is more stable.

### 3.1.2 Comparisons between different networks

A network is specified by its number of neurons, $N$, the matrix of its weights connections, $W$ and the activation function of its neurons.

Moreover, when computing the number of potential attractors of the network, it is possible to specify the following parameters: the quantization level $r$, the maximum
period, and the size of the hypersphere of robustness $\epsilon_{rob}$.

**Impact of the network’s size**

The impact of the network’s size is maybe the most important one. This is shown in Figure 4.4. The mean Lyapunov exponent is plotted in function of the number of attractors for networks of size $N = 1, 2, 4, 10, 50$ and $100$. The impact of the network’s size appears to be crucial both for the complexity and the number of attractors of the obtained networks.

![Figure 4.4](image)

**Figure 4.4:** These figures show the mean signal’s encoding capacity in RNN of different size. Each point represents the capacity of a matrix whose connections come from a uniform distribution ranging from -1 to 1. The network’s size appears as a determining parameter.

When increasing the network’s size from one neuron to two neurons and then to four neurons, better performance in terms of number of attractors of the mean signal is observed. In fact, the mathematical description of the one neuron has shown that only two kinds of dynamics occur: stable fixed points attractors and limit cycle attractors of size 2. For a quantization parameter $r$ set to 8, this gives a limited set of possible output strings (8 strings of size one, corresponding to fixed point attractors, and 28 strings of size 2, corresponding to period-2 limit cycle attractors). The number of different dynamics increases with the size, and consequently the number of attractors increases.

However, when the size continues to increase this leaning is inverted and for large size networks the number of attractors diminishes to disappear nearly completely.
This result is in accordance with the mathematical results obtained in the thermodynamic limit using the mean-field theory (Samuelides et al., 1999). The thermodynamic limit is the limit of huge size system (when $N$ tends to infinity).

This result can be explained intuitively: let us consider two neurons $i$ and $j$ connected through the weights $w_{ij}$ and $w_{ji}$. Since the potential and thus the state of the neuron $i$ is affected by the state of the neuron $j$ through the weight $w_{ij}$, the two neurons are dependant on each other. However, if the network is densely connected, when its size increases, the influence of one neuron to the other is more and more lost among the other connections. Neurons $i$ and $j$ look independent. This conjecture was proposed first by Amari and called “local chaos” (Amari, 1972). This denomination “local chaos” has to be differentiated from deterministic chaos. It is more related to Boltzmann’s hypothesis that, in a system of particles of huge size, the trajectories of the particles are independent and lead to statistical homogeneity. This “local chaos” is maintained and thus propagated through time leading to uncorrelated trajectories.

The fluctuations of the mean signal $m_N(t) = \frac{1}{N} \sum_{i=1}^{N} x_i(t)$ are equal, at each time step, to the mean of the fluctuations observed for each neuron. It appears that these fluctuations tend to compensate themselves. Thus, the standard deviation of the mean signal decreases, while for each individual neuron the standard deviation of their fluctuations remains constant. Since the mean signal tends to a stable value (a fixed point $m^*$), the encoding capacity decreases to $r$ (the number of equivalence classes). In fact, the mean signal appears as the pondered sum of $N$ random variables issued from the same distribution. The central limit theorem tells us that while the number of neurons increases, the standard deviation of the mean signal decreases as $1/\sqrt{N}$. This is proven experimentally in figure 4.5. This diminution of the mean signal’s potential attractors for large RNNs will lead us in Chapter 6 to consider the spatial dimension of the attractors.

Varying the network’s size also greatly affects the mean Lyapunov exponent: for networks of huge size, all networks, obtained from a random uniform distribution ranging from $-1$ to $1$, have the same global behavior characterized by a positive Lyapunov exponent (no variations appear in the figure). This means that the obtained networks are globally fully chaotic.

**Impact of the weight distribution**

Since our model works with frozen weights (the weights are determined once and then frozen), a network composed of $N$ neurons having a specified activation function is fully determined once the weight matrix is given.

In this chapter weights are chosen randomly (next chapters will discuss a different learning algorithm). However, there are various ways to randomize weights. Results plotted in Figures 4.6 and 4.7 show how the random weights distribution influences both the number of attractors and the mean Lyapunov exponent.

More specifically, a uniform distribution with variable ranges is used in Figure 4.6 while a normal distribution with variable mean and standard deviation is used in
Figure 4.5: These figures plot the evolution of the mean signal through time in RNN of different size. The standard deviation decreases in $1/\sqrt{N}$ while the size of the network is increased.

Figure 4.7.

Two observations can be made. Firstly, when the weights are small, the network’s activity is dampened and falls in a kind of lethargy: the number of attractors falls drastically. Secondly, modifying the center of the weight distribution impacts the stability of the obtained network to a large degree.

These results will serve as comparison when studying the impact of the modification of the weight distribution induced by a learning procedure.

*Impact of the activation function*

The state $x_i$ of a neuron $i$ results from the activation of its potential $u_i$ resulting from the influence of all the neurons (including $i$) and from external stimulus: $u_i = \sum_{j=1}^{N} w_{ij} x_j(n) + w_{i\theta} \varphi(n)$. The activation function of the neurons is given by the sigmoid function: $x_i = f(u_i) = 1/2(1 + \tanh(g u_i))$ where $g$ is the slope parameter of the sigmoid. Figure 4.8 shows the impact of the variation of this parameter. The value of this parameter seems to affect principally the mean Lyapunov exponent of networks.

In fact this parameter has been used by the team of Doyon et al. as a critical global parameter to lead the network from stable dynamics to chaotic dynamics (Doyon et al., 1993).

*Impact of the robustness required*
Figure 4.6: These figures compare the mean signal’s encoding capacity in a four neurons RNN for different connection’s matrix distribution. The weight distribution is uniform in a variable range. No robustness is assumed for the attractors.
Dynamical observations of random networks

Figure 4.7: These figures compare the mean signal’s encoding capacity in a four neurons RNN for different connection’s matrix distribution. The weight distribution is normal with different mean and standard deviation. No robustness is assumed for the attractors.

Figure 4.8: These figures compare the mean signal’s encoding capacity in a four neurons RNN when varying the slope parameter $g$ of the activation function (given by the sigmoid function $f(x) = \frac{1}{2}(1 + \tanh(gx))$). Random weights are given by a uniform distribution ranging from $[-2, 2]$. 
A further parameter is the robustness required for the attractors. An attractor is said to be \( \varepsilon \)-robust if the external stimulus can be modified in an hypersphere of size \( \varepsilon \) around its initial value without modifying the quantized symbolic output.

Figure 4.9 shows the impact of increasing the size of the robustness required for the attractors. As expected, the number of attractors decreases.

![Figure 4.9: These figures compare the mean signal’s encoding capacity in a four neurons RNN with random weights given by a uniform distribution ranging from ] 1, 1[ when varying the expected robustness of the attractors.](image)

### 3.1.3 weight distribution

The aim of this subsection is to continue to analyze the impact of the weight distributions. Figures 4.6 and 4.7 have already proven the strong influence of the shape of a distribution on the network’s underlying dynamics. Here, weight distribution of “special” networks (cf Figure 4.3) is analyzed: networks having a huge encoding potential and networks having a low mean Lyapunov exponent.

As before, weights are initialized randomly. Here, they are sample from an uniform distribution ranging from \(-1, 1\) is used. By computing the mean weight on a huge number of networks, we should have \(\mu(w_{ij}) = \mu(w_{ii}) = \mu(w_{is}) \rightarrow 0\). However, if the mean is performed only on a selected subset of all the random networks, this is no longer true. Table 4.1 gives the mean and the standard deviation obtained for two type of networks: networks having more than 150 potential attractors and networks having a mean Lyapunov lower than \(-4\). Each time the mean and the standard deviation are computed for the auto-connections \(w_{ii}\), the connections \(w_{ij} (i \neq j)\) and the stimulus connections \(w_{is}\). Nothing very relevant appears related to the connections \(w_{ij}\) and \(w_{is}\); means are nearly equal to zero and the two types of networks show more or less the same results. In contrast, different and characteristic results appear for auto-connections \(w_{ii}\): we show that networks having a high encoding capacity have globally negative auto-connections while positive auto-connections tend to stabilize the network.

These results are confirmed in Figures 4.10 and 4.11. Figure 4.10 shows an example of a weight distribution obtained for a network which has 178 potential attractors. All
auto-connections appear to be negative. Figure 4.11 shows an example of a weight distribution for a very stable network having only 10 potential attractors and having a mean Lyapunov exponent of $-4.9$. Three of the four auto-connections are positive, the fourth being only slightly negative.

Table 4.1: Mean and standard deviation of weights for two types of networks: for networks having an encoding potential greater than 150 and for networks having a mean Lyapunov exponent lower than $-4$.

<table>
<thead>
<tr>
<th></th>
<th>$\mu(w_{ij}) \pm \sigma(w_{ij})$</th>
<th>$\mu(w_{ii}) \pm \sigma(w_{ii})$</th>
<th>$\mu(w_{is}) \pm \sigma(w_{is})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>High capacity networks</td>
<td>-0.01 ± 0.56</td>
<td>-0.23 ± 0.49</td>
<td>0.06 ± 0.67</td>
</tr>
<tr>
<td>Low-mean Lyapunov networks</td>
<td>0.01 ± 0.61</td>
<td>0.31 ± 0.53</td>
<td>0.04 ± 0.57</td>
</tr>
</tbody>
</table>

Each subset is composed of 200 4-neuron RNNs. Each network is built from a uniform weight distribution in $[-1, 1]$, and must satisfy the corresponding condition.

Figure 4.10: Weight distribution of a network having 178 potential attractors and a mean Lyapunov exponent equals to $-0.49$.

Figure 4.11: Weight distribution of a network having 10 potential attractors and a mean Lyapunov exponent equalling $-4.9$. 

3.2 Qualitative Results

The previous subsection has shown that for all fully connected neural networks, a correlation exists between the mean Lyapunov exponent and the number of attractors after quantization: networks having a huge encoding potential are more “complex” and stable networks have a small number of potential attractors. Moreover, we have shown that these results remain true for all type of networks, whatever their size, their activation function or their weight distribution.

The mean Lyapunov exponent was obtained for a huge number of different inputs. Some of them fall in stable attractors and have negative Lyapunov exponents, others fall in chaotic attractors and have positive Lyapunov exponents, null Lyapunov exponents correspond to quasi-periodic attractors. In average, for a huge number of inputs, the mean Lyapunov exponent of small RNNs is always negative\(^1\). This indicates that the network is stable most of the time. However, chaotic regimes exist and the last section suggested that networks having a huge number of potential attractors have more chaotic attractors.

The aim of this section is to analyze whether the correlation between the number of attractors and the mean Lyapunov exponent of a network is not underlying another correlation, i.e. between the type of chaotic regimes encountered in a network and its encoding potential. However, the lack of rules or global framework for these analyses makes it very difficult. Therefore, only qualitative behaviors can be described.

First of all it has to be noted that small size RNNs are particularly adapted to provide different schematic types of chaos and roads to chaos\(^2\). The three roads to chaos described in Section 2.5 have been widely encountered. In fact these plots have been obtained from small RNNs manipulations. These three roads to chaos were the road to chaos by period doubling, the road to chaos by quasi-periodicity and the road leading to intermittency chaos.

From analyses of power spectrum, state diagrams and mean Lyapunov exponents for these three kinds of chaos, the following observations have been made:

- in chaos obtained from a period-2 bifurcation route, power spectra show one peak drowned in noise. At the limit, the power spectrum is similar to white noise, thus showing no structure. Mean Lyapunov exponents are huge (bigger than 0.2 in small RNNs). State diagrams show a nearly random signal (in accordance with the power spectrum).

- in chaos obtained from a quasi-periodic route, power spectra typically show multiple dominant incommensurate peaks and an approximatively white background noise three or four orders of magnitude below that of the dominant frequency. Mean Lyapunov exponents are tiny (about 0.005 to 0.01 in small RNNs). State diagrams bear the structure of a quasi-periodic signal slightly noised.

\(^1\)This does not remain true for networks of big size, as shown in Figure 4.4 where the mean Lyapunov exponent becomes positive for networks of size 50.

\(^2\)It will appear that this does not remain true for large RNNs.
• Power spectra of intermittency chaotic point are the most informative. Some periods clearly emerge from the white noise. Mean Lyapunov exponents are relatively small but not tiny. State diagrams usually bear the structure of nearby limit cycles attractors.

From return maps and power spectrum analyses, it became clear that when the encoding capacity of the studied network is huge, the prevailing chaos is more and more the intermittency chaos described in (Pomeau and Manneville, 1980), the result of a system at the edge of the saturation.

This chaos seems to originate from a frustration phenomenon that can be construed as a dynamical version of the hesitation of the network between possible responses to the presented situation. Ambiguous input leads to ambiguous dynamics (Kelso et al., 1995). Hence this chaos has been described as frustrated chaos (Bersini and Calenbuhr, 1997), (Bersini and Sener, 2002).

This type of chaos is represented in Figure 4.12: by moving from a period 3 attractor to a fixed point attractor, a chaotic regime occurs through changes in the external stimulus. Return maps of individual neurons are mostly saturated. This indicates the presence of a point-intermittent chaos. By analyzing the state diagram and the return maps, the period 3 cycle attractor remains visible in the chaotic window. By moving toward the fixed point attractor, the chaotic window increasingly indicates its presence.

This chaos is a dynamical regime appearing 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 appearances of these attractors. These successive appearances of almost stable regimes are separated by chaotic bursts. The relative duration of the dynamics into one of these temporary attractors depends on the respective size of its basin of attraction. All basins are intertwined in a fractal way.

4 Conclusion

This chapter has shown that small size fully connected RNNs can react in a multitude of discriminative ways to external stimuli based on the different dynamical attractors of the mean signal. The discrimination is based on a filter layer performing a quantization of temporal outputs of this mean signal into symbolic strings. This quantization restores the intuitive idea that two nearby inputs (in most cases) generate the same behavior and therefore this quantization enables the dealing with noisy inputs.

It was observed that for some weight values, chosen randomly, the encoding capacity (based on the number of potential attractors) can be huge (more than 300 for a 3-neurons RNN). This suggests that these small recurrent neural networks provided with a filter layer performing symbolic quantization of the mean signal, could be exploited to encode information through the cycles of the underlying dynamical system. For each input to be stored, the only rule is to have a periodic output: each encoded
Figure 4.12: Evolution of the dynamics while modifying the external signal from one stimulus corresponding to a period 3 attractor, to another one corresponding to a fixed point attractor. In-between chaotic dynamics appears. Signal, return maps and power spectra obtained from the mean signal $m_N(t)$ and return maps obtained from one neuron are plotted. $N = 5$
input generates a distinguished and dynamically stable symbolic output. In this way, inputs are stored in the attractors of the network’s internal dynamics.

By plotting the mean Lyapunov exponent of networks in function of their encoding capacity, the existence of a correlation between them has been demonstrated: networks having a huge encoding potential are more “complex” and stable networks have a small number of potential attractors. This result remains true for all type of networks, whatever their size, their activation function or their weight distributions. Auto-connections have shown to play a central role in this distribution: negative auto-connections tend to increase the encoding potential of a network while positive auto-connections stabilize the network.

In networks having a huge potential, chaos prevails more and more as a natural and spontaneous dynamical regime. Referring to Skarda and Freeman, this makes sense of the world (Skarda and Freeman, 1987). Basically, two roles are suggested for the chaos appearing in these neural networks: firstly, chaotic activity enables the rapid state transitions essential for information processing; secondly, from a cognitive point of view, chaos is essential for the creation of information (Skarda and Freeman, 1990a), (Kentridge, 2000), (Tsuda, 2001): chaotic behavior is necessitated by the brain in order to perform efficient non-deterministic symbolic computations during cognitive tasks.
Chapter 5

Learning using a gradient based algorithm

1 Introduction

In the last chapter we described how small size fully connected recurrent neural networks (RNNs) can encode a huge number of information in their dynamical attractors. Each piece of information feeds the network and is robustly associated with a symbolic string resulting from the quantization of the mean signal (the role of the quantization being to provide this robustness).

However, before being able to exploit their potential, a major problem remains to be solved: we are still working with fully connected random networks (networks having random connections). A priori there is no way to encode specific information: the mapping between a set of stimuli and their corresponding symbolic output is obtained a posteriori. A learning algorithm has to be found to learn the network to encode a specified set of information.

Different learning algorithms for neural networks can be found in the literature (see Section 2.4). Each of them has its specific field of application, i.e. its related learning tasks. Because of the inherent recurrences of recurrent neural networks, their internal representations (or state) are functions of both the external stimulus and the past representation. For this reason, these networks are most of the time used to map and to predict temporal series of data or to generate grammars (Kremer, 2001). Different types of algorithms have been proposed for this purpose. Principally gradient based algorithms, evolutionary algorithms and Hebbian algorithms can be distinguished. However, it seems that all these algorithms suffer drawbacks preventing their use in common life applications. These drawbacks may vary, but generally come from slow convergence (if the convergence has been guaranteed), expensive calculations, hard parameters tunings (demanding a lot of expertise from the user). Among others see (Werbos, 1990), (Feldkamp et al., 1998), (Jaeger, 2001).

To our knowledge, encoding information in dynamical attractors of a RNN is a learning task which has never been tested before. A learning algorithm has to be
found which adjusts connections in a way that after learning the network robustly associates a specified set of data to a specified set of symbolic strings. Since there are no prevailing algorithms for RNNs, the first algorithm tested is a gradient based algorithm. This choice has been made for two reasons. First, because of its influence in its feedforward version: thanks to the gradient descent based backpropagation algorithm, feedforward neural network have been successfully trained for different kinds of applications, going from classification to control. Second, because this type of algorithm has been widely used in the field of feedforward recurrent network (Kremer, 2001). Different versions of gradient based algorithms exist for RNNs. The “winner”, the famous backpropagation through time (BPTT) algorithm using an extended Kalman filter approach, has proven its quality for the prediction of temporal series. Here, the most basic version has been tested: the simple BPTT algorithm.

However, poor results were obtained: only small sets of data could be learned. Even the brute force random search gives better results in terms of convergence efficiency for some specified tasks. Moreover, dynamical analyses show that after learning the “chaoticity” of the “learned networks” dampens strongly.

2 Description of the learning task

The quantization of the mean signal introduced in 4.2.2 enables the robust storage of information. An information \( \mu \) being constituted by a pair composed of the external stimulus \( x_L^\mu \) and the symbolic periodic string \( S_L^\mu(p, r) \) (of size less or equal to \( p \), and composed of letters belonging to the alphabet \( A_r \)) generated by the quantized mean signal. The learning task consists of storing a set of information in the network:

\[
\mathcal{D}_L = \{ \mathcal{D}_1^L, \ldots, \mathcal{D}_q^L \} \quad \text{with} \quad \mathcal{D}_i^\mu = \{ x_L^\mu, S_L^\mu(p, r) \} \tag{5.1}
\]

where \( L \) is set for learning. The learning procedure has to adjust the network’s parameters in such a way that after learning, an external stimulus learned brings the network’s dynamics in a specific attractor, generating the expected symbolic string in its mean signal. Since both the external stimulus and the output string are fully specified, this learning task necessitates supervised learning algorithms.

3 Description of the BPTT algorithm

3.1 Introduction

Gradient-based learning algorithms have been already widely described and discussed in the literature (among others (Rumelhart et al., 1986) and (Williams and Zipser, 1990)). The network learns through a smooth gradient-based learning process that slightly modifies its weights as a feedback function of the network’s output. However, because each technique has its variants, and each model needs adaptations, it is worth specifying the equations used here.
The feedforward backpropagation algorithm cannot be directly transferred to RNNs, because the backpropagation pass presupposes that the connections between the units induce a cycle-free ordering. The solution of the BPTT approach is to “unfold” the network in time, by stacking identical copies of the RNN, and redirecting connections within the network to obtain connections between subsequent copies. This gives a feedforward network which is amenable to the backpropagation algorithm (Werbos, 1990).

Before giving the equation used, let us first review the process of the simple backpropagation algorithm.

### 3.2 The backpropagation algorithm

#### 3.2.1 Introduction

In order to train single layer neural networks, a simple gradient descent method can be applied. This method is based on the computation of the error appearing on output neurons between observed values and expected values. The gradient of this error function is computed with respect to each weight. It tells us how a small change in that weight will affect the overall error. We can then subtract a small proportion $\epsilon$ (called the learning rate) of the gradient from the weights to perform gradient descent.

However, it has been shown that single layer perceptron can only fit linear functions (Minsky and Papert, 1969). The addition of hidden layers allows the perceptron to fit any functions with any accuracy (Cybenko, 1989). These perceptrons are called multi-layer perceptrons. Figure 5.1 shows a very classical multi-layer perceptron composed of one input layer, one output layer and one hidden layer.

![Figure 5.1: A very classical 1-layered perceptron.](image)

Adding hidden-layers is not without a cost, the gradient descent method is no longer sufficient by itself. Indeed, gradient descent is based on the computation of the error on output neurons. For multi-layered perceptrons, not only output neurons exist but also hidden neurons. No expected output values exist for these neurons.

The idea of Rosenblatt (Rosenblatt, 1962) was to back-propagate the error from the output layer to the hidden layer (and then to continue backpropagation until the input layer. Therefore it is possible to apply the gradient descent in these layers too. The first gradient descent approach for training multilayered nets was done by Amari in 1967 (Amari, 1967) (he used a single hidden layer perceptron to perform nonlinear classification).
In principle, backpropagation provides a way to train networks with any number of hidden units arranged in any number of layers. (There are clear practical limits, which we will discuss later.) In fact, the network does not have to be organized in layers - any pattern of connectivity that permits a partial ordering of the nodes from input to output is allowed. In other words, there must be a way to order the units such that all connections go from “earlier” (closer to the input) to “later” ones (closer to the output). This is equivalent to stating that their connection pattern must not contain any cycles. Networks that respect this constraint are called feedforward networks; their connection pattern forms a directed acyclic graph.

3.2.2 Learning Algorithm

We want to train a multi-layer feedforward network by gradient descent to approximate an unknown function, based on some training data consisting of pairs \((\bar{\mathbf{\iota}}, \bar{\mathbf{o}})\). The vector \(\bar{\mathbf{\iota}}\) represents a pattern of inputs to the network, and the vector \(\bar{\mathbf{o}}\) the corresponding targets (desired output) as described in the model. The overall gradient with respect to the entire training set is just the sum of the gradients for each pattern. Below we will therefore describe how to compute the gradient for just a single training pattern.

<table>
<thead>
<tr>
<th>Table 5.1: Definitions used for the backpropagation algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>(net_i = \sum_{j=1}^{n} w_{ij} x_j)</td>
</tr>
<tr>
<td>(x_j = f(net_j))</td>
</tr>
<tr>
<td>(\delta_j = -\frac{\partial E}{\partial net_j})</td>
</tr>
<tr>
<td>(\Delta w_{ij} = -\varepsilon \frac{\partial E}{\partial w_{ij}})</td>
</tr>
<tr>
<td>(\varepsilon)</td>
</tr>
<tr>
<td>(A_i = {j : \exists w_{ij}})</td>
</tr>
<tr>
<td>(P_j = {i : \exists w_{ij}})</td>
</tr>
</tbody>
</table>

As for linear networks, we can expand the gradient into two factors by using the chain rule:

\[
\Delta w_{ij} = -\varepsilon \frac{\partial E}{\partial net_j} \frac{\partial net_i}{\partial w_{ij}} \tag{5.2}
\]

The first factor is the error of unit \(i\), and since we know that the network is feedforward, the computation of the second factor is straightforward:

\[
net_i = \sum_{j \in A_i} w_{ij} x_j(t)
\]

\[
\frac{\partial net_i}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \sum_{k \in A_i} w_{ik} x_k = x_j
\]
Equation 5.2 becomes:

$$\Delta w_{ij} = \varepsilon \delta_i x_j$$  \hspace{1cm} (5.3)$$

To compute this gradient, we thus need to know the activity \( f \) and the error signal \( \delta \) for all relevant nodes in the network.

Assuming we are using sum-squared loss function

$$E = \frac{1}{2} \sum_o (o_o - x_o)^2$$

since \( x_o = f(\text{net}_o) \), the error signal for the output unit \( o \) is:

$$\delta_o = - \frac{\partial E}{\partial \text{net}_o} \frac{\partial f(\text{net}_o)}{\partial \text{net}_o} = (o_o - x_o).f'_o(\text{net}_o)$$

For hidden units, we must back propagate the error from the output nodes (hence the name of the algorithm). By using the chain rule, we can expand the error of a hidden unit in terms of its posterior nodes:

$$\delta_j = \sum_{i \in P_j} \frac{\partial E}{\partial \text{net}_i} \frac{\partial \text{net}_i}{\partial x_j} \frac{\partial x_j}{\partial \text{net}_j}$$

Of the three factors inside the sum, the first represents \( \delta_i \), the error signal of node \( i \) which is back-propagated. The second is

$$\frac{\partial \text{net}_i}{\partial x_j} = \frac{\partial}{\partial x_j} \sum_{k \in A_i} w_{ik} x_k = w_{ij}$$

while the third is the derivative of node \( j \)'s activation function. Since the activation function is a sigmoid, we have:

$$\frac{\partial x_j}{\partial \text{net}_j} = \frac{\partial f_j(\text{net}_j)}{\partial \text{net}_j} = f'_j(\text{net}_j) = f_j(1 - f_j)$$

Putting all the pieces together we get:

$$\delta_j = f_j(1 - f_j) \sum_{i \in P_j} \delta_i w_{ij}$$

After having back-propagated the error, weights are adjusted with a learning rate \( \varepsilon \):

$$\Delta w_{ij} = \varepsilon \delta_i x_j$$  \hspace{1cm} (5.4)$$
with the error signals given by:

\[ \delta_o = (o_o - x_o)x_o(1 - x_o) \text{ for the output units} \]
\[ \delta_j = x_j(1 - x_j) \sum_{i \in P_j} \delta_i w_{ij} \text{ for the hidden units} \]  \hspace{1cm} (5.5)

### 3.3 Backpropagation Through Time Algorithm

The first step in this algorithm consists of unfolding the recurrent neural network by stacking identical copies of it, and redirecting connections within the network to obtain connections between subsequent copies. This process of unfolding a network through time is represented in Figure 5.2.

The teacher data is given in Equation 5.1 and consists of the multiple pairs \( D^\mu_L \) constituted by the external stimulus and the symbolic output string. The computational steps are the following:

- **Forward pass:** All stacked network are updated, from the first copy \((\text{time}=1)\), working upwards through the stack. At each time \( n \), the state \( x(n) \) for all the neurons is computed from \( \iota \) and \( x(n-1) \).

- The error propagation term \( \delta_i(n) \) is computed for each time \( n \) and neuron \( i \), by proceeding backward through \( n = T, \ldots, 1 \) (eq: 5.6). For convenience reasons, \( \delta_i(T + 1) \) is set to 0.

\[ \delta_i(t) = f'_i \left[ \sum_k \delta_k(t + 1)w_{ki} + (o_i - x_i) \right] \text{ if } i=\text{output unit} \]
\[ \delta_i(t) = f'_i \left[ \sum_k \delta_k(t + 1)w_{ki} \right] \text{ if } i \neq \text{output unit} \] \hspace{1cm} (5.6)

- The weight connections are adjusted according to:

\[ \Delta w_{ij} = \varepsilon \sum_{t=1}^{T} \delta_i(t)x_j(t - 1) \] \hspace{1cm} (5.7)
\[ \Delta w_{\text{bias}} = \varepsilon \sum_{t=1}^{T} \delta_i(t)o_j(t) \] \hspace{1cm} (5.8)

After each weight transformation, the network is simulated during some time steps in order to get out of the transient effects (around 10 time steps). After 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.

### 3.4 Learning stimulus: practical considerations

The learning task consists of the hetero-association between external stimuli and quantized symbolic strings obtained from the mean signal. We have shown in Section 3.3.2
that the mean signal can be obtained from an additional neuron having a mean activation function. Figure 5.2 shows the unfolding process of this network. The network receives a constant stimulus \( \iota \in \mathbb{R}^3 \) and the mean signal \( m_n(t) \) is observed in the output.

![Unfolding process of a 4 neurons network composed of a 3 neurons RNN and a fourth meaner neuron.](image)

Learning this network using BPTT requires specific attention when computing the errors \( \delta_i(t) \) (Equation 5.6). The derivation of the activation function for the mean neuron is of course different than the one of the other neurons (its derivative is equal to \( 1/N \) where \( N \) is the number of neurons).

To improve the learning, two important modifications have been performed:

- Before learning, weights are randomly initialized with small values from an uniform distribution;
- From each data to learn, numerous noised copies are performed and added in the data set;

### 3.5 Alternatives to the classical BPTT algorithm

#### 3.5.1 Real-time recurrent learning

The real-time recurrent learning (RTRL) is a gradient-descent method which computes the exact error gradient at every time step and therefore suitable for online learning tasks. A lot can be found about RTRL in (Doya, 1995) (Williams and Zipser, 1989). RTRL is mathematically transparent and in principle suitable for online training. However, the computational cost is \( O((N + L)^4) \) for each update step, because we have to solve the \((N + L)\)-dimensional system for each weight, where:

- \( N \) is the number of internal neurons
- \( L \) is the number of output neurons

---

1These modifications have been performed thanks to advices from D.Prokhorov, world-famous expert of the BPTT algorithm and more particularly of the Extended Kalman Filter approach.
This high computational cost makes RTRL useful for online adaptation only when a very small network suffices.

### 3.5.2 Extended Kalman filter

Pure gradient-descent techniques for optimization generally suffer from slow convergence when the curvature of the error surfaces is different in different directions. In that situation, on the one hand, the learning rate must be chosen small to avoid instability in the directions of high curvature. On the other hand, this small learning rate may lead to unacceptably slow convergence in the direction of low curvature (see Figure 5.3). The incorporation of curvature information into the gradient-descent process is a general remedy. This requires the calculation of the second-order derivatives, for which several approximative techniques have been proposed in the context of convergence, especially near an optimum where the error surface can be reasonably approximated by a quadratic function. Dos Santos & von Zuben (2000) and Schraudolph (2002) provide references, discussion, and propose approximation techniques which are faster than naive calculations.

Among them, the extended Kalman Filter (EKF) is a state estimation technique for nonlinear systems derived by linearizing the well-known linear-systems Kalman filter around the current state estimate.

Currently, the best results in RNN training are achieved with EKF (demonstrated especially in many remarkable achievements from Lee Feldkamp’s research group (1994, 1998).

#### 4 Numerical Results

The problem consists in finding a small size RNN which can code all the data $D_{\mu}^L (\mu = 1, \ldots, q)$. Each data being composed of an external stimulus and its corresponding symbolic string. The first part of this section tests the BPTT learning algorithm for different data sets in term of time efficiency. Dynamical observations of the learned networks will be performed in the next section. The last section focuses on matrix analyses.
4.1 Efficiency of the BPTT

Since encoding patterns in dynamical attractors of RNNs has never been tested before, there are no possible points of comparison with any other algorithm. The efficiency of the BPTT algorithm could only be estimated in function of the success or the failure of the given data set to learn, hence by computing its limits.

However, the BPTT is based on random weight initializations: if the algorithm fails to converge, weights are re-initialized randomly. Since the potential of random RNN can be huge, it is possible to imagine that after some random initializations the network falls in a configuration fulfilling the specified learning task. Due to this random mechanism, it is possible to learn with BPTT without relying on the gradient process. However, to randomly find a good network can take a lot of time.

With the notion of time appearing, one may ask oneself whether it is possible to learn a specified data set in a feasible time. Figure 5.4 compares relative time efficiency of the BPTT learning algorithm for various data sets (blue crosses) in a four neurons RNN. Random initializations of the network are performed using an uniform distribution ranging from $-0.1$ to $0.1$. The left figure shows that the computational time increases exponentially while the number of stimuli, encoded in fixed point attractors of one output neuron, increases. Moreover, BPTT fails to encode more than 7 data (at least in a feasible time). The right figure shows that the BPTT algorithm has difficulties in associating stimuli with limit cycles attractors(it fails to encode one stimulus in a size-4 cycle). These two plots are compared with brute random search (red circles) which is the only other available “algorithm. Stunningly, random search gives comparable results, sometimes worse (random search fails to store more than 5 stimuli in fixed points attractors) but sometimes better: for example, for the finding of hetero-association between one stimulus and a size-4 cycles.

The BPTT algorithm can be tuned using a huge number of parameters (among others, the learning rate parameter, the weight distribution after a random initialization, the unfolding size, etc.). Modifying these parameters can change its performance for a given task drastically. For example, while learning to encode stimuli in fixed point attractors, increasing the range of the random weight distribution by a factor ten increases the computational time by the same factor. The reverse is true when learning cycles. BPTT appears as a difficult parameter tuning problem, demanding a lot of expertise from the user. The same observations are made when using BPTT for other learning tasks (Jaeger, 2001) (or from direct conversations with D.Prokhorov).

A remark to the advantage of the BPTT algorithm must be given. Same results have been obtained when supposing that stimuli are no longer encoded in the quantized output signal of one neuron but in quantized output signals of multiple neurons. On the contrary, this modification drastically increases the difficulty for a brute force random search: it becomes difficult to find a network which associates one stimulus in a 3-dimensional fixed point attractors. Thus, BPTT radically outperforms random search when the spatial configuration of attractors is specified, and not only their temporal and global behavior through the mean signal.
Until here, we have shown that when fed by static stimuli, BPTT is more suited to learn static outputs which could be spatially specified.

### 4.2 Dynamics of network after learning

Learning a specified data set in a RNN through the BPTT learning algorithm modifies its weight distribution in a configuration in such a way that when the network receives an encoded stimulus, its internal dynamics converge to the specified attractor. In this subsection, analyses are performed on dynamics occurring when the network is fed by unlearned stimuli.

In Figure 5.5 mean Lyapunov exponents of learned networks are plotted in function of their total capacity $^2$. These results are compared with results obtained from networks “learned” through a brute force random search. These two figures can be related to Figures 4.6 obtained for random networks using an uniform weight distribution. This is explained by the fact that BPTT relies on random weight initializations (here performed using an uniform distribution ranging from $-0.1$ to $0.1$) and brute force random search is, of course, based on random weight initializations (here performed using an uniform distribution ranging from $-1$ to $1$).

From Figure 5.5, it appears that BPTT needs very stable networks to converge. This can be explained intuitively in the following way. Let us imagine that we have to associate one stimulus with a fixed point attractor. Let us also suppose that after some

---

$^2$The encoding capacity is computed by testing the resulting dynamics for 1000 different external stimuli. An external stimulus $i$ is stored in $D$, if the corresponding quantized output is a cycle of period $\leq p$ (with $p = 4$) which has not been previously obtained. The quantization level $r$ is set to 8.
Figure 5.5: Total encoding capacity and mean Lyapunov exponent of RNNs which have learned to associate 3 stimuli to distinct fixed points attractors. The left figure shows the results for RNNs learned through the BPTT algorithm, while the networks in right figure have been found through a brute force random search. The BPTT search strongly dampens the resulting network capacity. \( N = 3 \)

random weight initializations we are not so far from the expected behavior. The error is computed and back-propagated through the layers of the unfolded network. Then weights are slightly modified to get closer to the expected behavior. However, instead of getting closer, these modifications may provoke a bifurcation or may lead to a cusp catastrophe preventing any further convergence (see Figure 5.6). Only a new weight randomization can overcome this problem.

Since networks having a huge number of attractors have a huge number of bifurcations, there is only a small chance to obtain such networks after BPTT learning. This also explains the difficulty to learn to associate stimuli with limit cycles attractors: it is difficult to provoke an expected bifurcation using a gradient based algorithm. Such data may be “learned” uniquely using the random mechanism inherent to BPTT learning and possibly small weight adjustments using the back-propagated errors.

### 4.3 Weight distributions after learning

The BPTT learning algorithm relies on an initial weight randomization followed by slight weight modifications according to back-propagated errors’ values. Assuming an initial uniform random distribution ranging from -0.1 to 0.1, the learning procedure modifies it. The more time it takes to learn, the more the weight distribution obtained after learning will be transformed.

Figures 5.7 show how the learning procedure modifies weight distributions: by constraining the network to learn more, weight distributions are continuously enlarged. However, the connections \( w_{ij} \) and the auto-connections \( w_{ii} \) follow different trends: \( \mu(w_{ij}) \) decreases continuously, whereas, \( \mu(w_{ii}) \) increases. Auto-connections
Learning using a gradient based algorithm

Figure 5.6: This figure shows how bifurcations prevents any convergence.

appear to be globally positive corroborating results obtained in previous chapters where it was shown that stable networks have positive auto-connections (see table 4.1).

It has to be noted that learning 7 stimuli in fixed point attractors induces different trends. This can be explained by the fact that this data set is so hard to learn that the learning procedure relies more on a special predisposition provided by a specific weight randomization than on the gradient descent mechanism by itself. This implies that the number of randomization required increases drastically (from 1, for learning 6 stimuli, to several dozens for 7 stimuli).

Figure 5.7: Global properties $\mu$ and $\sigma$ of RNNs’ weight distributions $w_{ij}$, $w_{ii}$ and $w_{is}$ after learning an increasing size of stimuli in fixed point attractors. Each time mean values surrounded by standard deviations ($\mu + \sigma$ and $\mu - \sigma$) are plotted. $N = 3$

These statistics are corroborated by the direct observations of weight distributions on learned networks, as appearing in Figures 5.8 and 5.9. In the first figure the data set is composed of 2 stimuli while in the second figure the data set is composed of 6 stimuli. The same trends are observed: all the distributions are enlarged when the size of the data set is increased and the distribution of the auto-connections appears
globally more and more positive.

Figure 5.8: Example of a weight distribution of a RNN which has learned 2 stimuli in fixed point attractors after BPTT. \( N = 3 \)

Figure 5.9: Example of a weight distribution of a RNN which has learned 6 stimuli in fixed point attractors after BPTT. \( N = 3 \)

5 Conclusion

The very classical backpropagation through time (BPTT) learning algorithm has been introduced in this chapter. It is an adaptation of the backpropagation algorithm for recurrent neural networks. No convergence proof is given and hence no guaranty exists that the network can learn a desired set.

This BPTT algorithm has been tested in order to give a straightforward way of encoding specified stimuli in quantized symbolic strings of the mean signal of RNNs. After learning, when fed by a specific stimulus, the network converges to a limit cycle or a fixed point attractor of the inner dynamics having the required mean signal.

This algorithm showed poor results in term of efficiency due to the fact that the gradient descent based approach is often stopped by the numerous bifurcations encountered during the descent (Doya, 1992), (Bengio et al., 1994). As a consequence, learned networks appear entirely stable with a weak number of potential attractors.
This can be explained by the fact that high encoding capacity networks have complex internal dynamics which are incompatible with smooth weight variations.

Because recurrent networks are inherently dynamics, trying to make a static use of them is not so easy. Applying static memory learning procedures to them is not a common way to work. Most of the time they are used for time series prediction or other temporal applications.

However, for both procedures, it appears difficult to encode many external stimuli in specified cycles. And because of the non-linear and non-continuous nature of the output, it seems difficult to imagine an efficient gradient based learning algorithm. In addition, the use of these RNNs seems compromised.

In order to use these small RNNs, another viewpoint has to be adopted. We think that instead of specifying the cycles we want to learn, we could imagine using these networks in a heteronomous way. These small chaotic RNNs could be used as powerful primitive building blocks in more complex architectures. As such the learning procedure is no longer to be found inside these RNNs, but on the connections between them. In this perspective, the cycles of the small RNN could be seen as primary symbols on which we have no influence, except on the way we use them. Hence, we could work with small RNN as basic random building blocks. This gives us the possibility, when facing a problem, to keep a building block or to throw it out. The cycles of the small RNN could be seen as primary symbols on which we have no influence, except on the way we use them. These small RNN could be part, as neuro-modules, of higher architectures, on which learning algorithms could be applied.

Another viewpoint, followed in this thesis, is to test other types of learning algorithms.
Chapter 6

Learning using an iterative supervised Hebbian algorithm

1 Introduction

Chapter 4 has explained how small size fully connected RNNs with randomly tuned connections allow the storage of a huge number of static and cyclic attractors. Since the activation function of neurons is continuous, robustness is provided by a symbolic quantization of the output space.

However, at the same time, the lack of a learning algorithm compromises any working application. Storing capacity was investigated in an abrupt mechanism (brute force random search): no way was found to store a predefined set of patterns in the attractors. A first attempt to reach this goal has been described in chapter 5: the classical backpropagation-through-time algorithm was tested to store the desired information in these networks.

During the tests, learning gave very disappointing results. All the inherent storing capacity of learned networks was lost and only a tiny number of patterns led to successful learning. This gradient-based algorithm showed difficulties to overcome, by smooth weight variations, the catastrophic bifurcations occurring. By analyzing the weight distributions of learned networks, it appeared that learning tends to drive weight to regions favoring very stable networks and thus impoverishing the information capacity all together.

This chapter and the following ones focus on the implementation and the testing of another kind of learning algorithm, based on the classical Hebbian rule. Already in 1982, Hopfield developed a simple algorithm based on the Hebbian prescription rule to store patterns in fixed point attractors of RNNs (Hopfield, 1982). During more than ten years, this seminal work has been inspiring a huge community of researchers. A lot of work has been dedicated both to the development and the improvement of the Hebbian algorithms and to the computation of theoretical limits of RNNs. All the authors (like ourselves) were convinced that recurrent neural networks are the most promising machines for storing and retrieving information (for a deep overview of the
state of the art at that time, see (Domany et al., 1995) and (Hertz et al., 1991)).

At that time, the learning task commonly used was defined as the storage of a set of static patterns in the network’s fixed point attractors, as memories; after learning, the network was able to recover a learned pattern from a noisy pattern. The network worked as a content-addressable memory.

In our work, following the ideas mentioned in previous chapters, the learning task was slightly modified. Firstly, we believe that not only static fixed point attractors but also limit cycles attractors should be used in the storing process. Secondly, external stimuli are being used as part of the information to drive the network to other dynamics. This helps both the recovery of noisy patterns and the increase of the storing capacity.

Let us enlighten these two points by using a simple example. Very intuitively, imagine a two neurons network where neurons can only have the two values: \{-1, 1\}. Only four fixed point attractors can be exploited. By adding cycles of size two, this number grows to 16. Cycles of length two are:

\[(+1, +1)(+1, -1) (+1, +1)(-1, +1) (+1, +1)(-1, -1)\]
\n\[(+1, -1)(-1, +1) (+1, -1)(-1, -1) (-1, +1)(-1, -1)\]

This growth in the number of potential attractors justifies the use of cycles. External stimuli have two roles. Firstly, they widen the possible dynamics crossed by the network. This increases the number of attractors practically exploitable. Secondly, it appears in the example above that multiple cycles share common patterns (for example, 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 and additional information.

The aim of this chapter will be to adapt existing iterative Hebbian algorithms to the proposed learning task. The next section focuses on the definition of the learning task and on the subsequent model’s modification to suit to it. Section 2.3.1 describes the version of the Hebbian learning that our neural machine relies on. The impact of the modifications with respect to the original Hopfield proposal will be analyzed quantitatively and qualitatively in Section 3.

2 Description of the learning task and of the model

There is a key difference between the gradient based algorithms defined in the previous chapter and the Hebbian algorithms: while the former are global, Hebbian algorithms are local. This means that each weight \(w_{ij}\) is adjusted in function of values of its direct neighbors which here are the states of neuron \(i\) and \(j\).

The learning task defined in the previous chapters was to find a network which maps specific stimuli to symbolic strings obtained from the quantization of the mean signal. The BPTT algorithm enabled such findings (but unfortunately with poor results) due to its global mechanism: the error observed in the output was distributed
among all neurons. Applying a Hebbian algorithm to learn this mapping is no more workable: it will only result in the learning of the 3 weight connections impinging the meaner neuron (these connections are visible Figure 3.2). To get all weights modified by the learning process, constraints have to be specified on all neurons. To achieve this, stimuli are learned in the attractors’ spatio-temporal configurations.

2.1 The learning task

The network is used to store a set of \( q \) data:

\[
D = \{D^1, \ldots, D^q\}
\]

(6.1)

The number of data stored in the network is quantified by a load parameter \( \alpha = q/N \). Each data \( D^\mu \) is defined by a pair composed of:

- a pattern \( \chi^\mu \) corresponding to the external stimulus feeding the network;
- a sequence of patterns \( \varsigma^{\mu,i}, i = 1, \ldots, l^\mu \) to store in a dynamical attractor.

\[
D^\mu = (\chi^\mu, (\varsigma^{\mu,1}, \ldots, \varsigma^{\mu,l^\mu})) \quad \mu = 1, \ldots, q
\]

(6.2)

where \( l^\mu \) is the size of the sequence \( \mu \) and may vary from one data to another. Each pattern \( \mu \) is defined by assigning digital values to all neurons:

\[
\begin{align*}
\chi^\mu &= \{\chi^\mu_i, i = 1, \ldots, N\} \quad \text{with} \quad \chi^\mu_i \in \{-1, 1\} \\
\varsigma^{\mu,k} &= \{\varsigma^{\mu,k}_i, i = 1, \ldots, N\} \quad \text{with} \quad \varsigma^{\mu,k}_i \in \{-1, 1\}
\end{align*}
\]

(6.3)

By suppressing the external stimuli and by defining all the sequences’ sizes \( l^\mu \) to 1, this task is reduced to the classical learning task originally proposed by Hopfield: the storing of pattern in the fixed point attractors of the underlying RNN 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: \( \xi^\mu \).

Figure 6.1 represents a sequence of size 2 unfolding in a network of four neurons. The external stimulus feeding the network appears in the upper part (a) of the figure.

Before testing the learning of sequences, our algorithm has first been validated by storing sets of static patterns in RNNs, as in the original Hopfield model. Thus, the data set becomes:

\[
D^\mu = \xi^\mu \quad \mu = 1, \ldots, q
\]

(6.4)

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 a duplicated information:

\[
D^\mu = (\xi^\mu, \xi^\mu) \quad \mu = 1, \ldots, q
\]

(6.5)
then, the storing of hetero-associative data is tested:

\[ D^\mu = (\chi^\mu, \xi^\mu) \quad \mu = 1, \ldots, q \]  

(6.6)

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

2.2 The model

The model used is described in section 3.3.2: the network is fully connected, the temporal update rule is discrete. While storing static patterns or fixed points, it can indifferently be asynchronous or synchronous. This is no longer the case when storing sequences or cycles, for which the updating necessarily must be synchronous: it is the global activity due to one pattern that generates the next one.

However, it is worthwhile mentioning a quantitative difference: while in previous chapters networks were kept small to ease the understanding of the obtained dynamics, here we deal with hundreds of neurons (which is still tiny compared to the brain). This change of scale is the result of the learning task: patterns are no longer stored temporally only, but also spatially. Moreover, the stored patterns have binary values (the quantization level is set to 2) which reduces the set of possibilities.

With regard to the neurons’ activation functions, this chapter uses the classic sign function. This allows to conveniently work with bit-patterns: the set of values for the internal state is \([-1, 1]\) and allows easy comparison with expected stored bit-patterns. Using this activation function, the algorithm’s performance in terms of its encoding capacity and the stored patterns’ noise robustness will be computed. The next chapter will focus on dynamical investigations on learned networks.
2 Description of the learning task and of the model

2.3 Description of the algorithm

2.3.1 Hopfield’s auto associative model

In the basic Hopfield model (Hopfield, 1982), all connections need to be symmetric and no auto-connection can exist. Hopfield has proven that these constraints are sufficient to define a Lyapunov function $H$ for the system:

$$H = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} x_i x_j \quad (6.7)$$

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. 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 defined in Equation 6.4 can be stored in the network by using the following Hebbian learning rule, obviously respecting the Hopfield model’s constraints (symmetric connections and no auto-connection):

$$w_{ij} = \frac{1}{N} \sum_{\mu=1}^{P} \xi^\mu_i \xi^\mu_j \quad w_{ii} = 0 \quad (6.8)$$

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. (1985 and 1987) by relying on a mean-field approach and on replica methods originally developed for spin-glass models. Their theoretical results show that such type of networks, when coupled with this learning rule, is unlikely to store more than $0.14 N$ uncorrelated random patterns.

2.4 The perceptron version of the Hebbian learning

A better way of storing patterns is inspired by the perceptron learning rule (for a detailed description of this algorithm see van Hemmen and Kuhn, 1995 and Forrest and Wallace, 1995). The principle of this algorithm is as follows: at each learning iteration, the stability of every nominal pattern $\xi^\mu$, is tested. Whenever one pattern has not reached stability yet, the responsible neuron $i$ sees its connectivity

1A Lyapunov function defines a lower bounded function whose derivative is decreasing in time
Learning using an iterative supervised Hebbian algorithm

reinforced by adding a Hebbian term to all the synaptic connections impinging on it:

\[ w_{ij} \mapsto w_{ij} + \epsilon \mathbf{\xi}_i \mathbf{\xi}_j \]  

(6.9)

where \( \epsilon \) defines the learning rate. All patterns to be learned are repeatedly tested for stability and, once all stable, the learning is complete.

2.4.1 First enhancement (Forrest and Wallace): training with noise

Whereas this algorithm has been shown to enhance the capacity of the network for perfect storage, nothing guarantees the robustness of the attractors. When patterns are stored in the dynamics’ fixed point attractors and when \( \alpha \) is increased to 0.5, the basins of attraction become negligibly small (Forrest and Wallace, 1995). A slight modification of the internal state of one neuron and, consequently, the network may bifurcate into another attractor.

In order to not only store the patterns, but also to ensure a sufficient enough content-addressability, we must 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 in this work.

A learning noise parameter \( ln \) is defined to tune the learning phase. A noisy pattern \( \mathbf{\xi}_{ln} \) is obtained from a pattern to learn \( \mathbf{\xi} \) by choosing a set of \( ln \) items, randomly chosen among all the initial pattern’s items, and by switching their sign. Thus, \( d_H(ln) \) defines the Hamming distance between the two patterns:

\[ d_H(ln) = \sum_{i=1}^{N} d_i \quad \text{where} \quad \begin{cases} d_i = 0 & \text{if } \xi_i^\mu \xi_{i,ln}^\mu = 1 \quad \text{(items equals)} \\ d_i = 1 & \text{if } \xi_i^\mu \xi_{i,ln}^\mu = -1 \quad \text{(items differents)} \end{cases} \]  

(6.10)

However, to ease the reading, Hamming distances will be always normalized to 100. For example in a network of 25 neurons, two opposite patterns have an Hamming distance equal to 25 which gives 100 after normalization.

2.4.2 Second enhancement (Molter et al.): using external stimuli

External stimulus is one of the additions proposed here to increase the number of learned patterns and to help their recovery. 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...
3 Experiments and Results

learned by adopting the same approach as given in Equation 6.9. 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, including the connection coming from its external stimuli:

\[
    w_{is} \mapsto w_{is} + \varepsilon_b \chi_i^\mu \xi_i^\mu
\]  

(6.11)

where \( \varepsilon_b \) defines the learning rate applied on the external stimulus’ connections and may differ from \( \varepsilon \).

As a consequence, the number of parameters increases. We now have two noise parameters to tune during the learning phase: the noise imposed on the internal states \( \ln \) and the noise imposed on the external stimulus \( \ln_b \).

2.5 Learning cycles

The learning rule defined in Equation (6.9) naturally leads to asymmetrical values for the weights. It is no longer possible to define a Lyapunov function for this system, the main consequence being the inclusion of cycles in the set of “memory bags”.

As for fixed points, the network can be trained to converge to such limit cycles attractors by modifying the Equation (6.9): weight \( w_{ij} \) of the connection going from neuron \( j \) to neuron \( i \) is modified according to the expected value of neuron \( i \) at time \( t+1 \) and the expected value of neuron \( j \) at time \( t \) (see Equation 6.13 in Table 6.1).

The pseudo-code of the algorithm used to store cycles is described in Table 6.1 (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 (6.14) in Table 6.1 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, 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 Experiments and Results

Two types of experiments have been conducted aiming at assessing both the storing capacity and the robustness to noise. It has to be noted that this noise robustness is not only described by the content addressability of the pattern to recover. Indeed, data is composed of two pieces of information: the external stimulus and the limit cycle attractor. Hence, we are not recovering the stored pattern (or cycle) only on the basis of a part of its content, but also the external stimuli have to be partly correct.
100 Learning using an iterative supervised Hebbian algorithm

Table 6.1: Pseudo code of the iterative supervised Hebbian algorithm

1. Fill the data set \( D \) with the data to learn;
2. A data \( D^\mu \) is chosen randomly from the data set \( D \);

**learning with no noise**

3. For \( \text{idcyc} \) going from 1 to \( l_\mu \):
   
   (a) The external stimulus is initialized with \( \chi^\mu \)
   
   (b) Internal states are initialized with \( \varsigma^\mu \text{idcyc} \).
   
   (c) For each neuron \( i \), the equality:

\[
\text{sign}(x_i) \overset{?}{=} \varsigma^\mu \text{idcyc+1} \mod l_\mu
\]

is tested. If they are not equal,

- all the synaptic connections impinging on this neuron are modified according to the Hebbian learning rule:

\[
\begin{align*}
\text{wij} & \mapsto \text{wij} + \varepsilon \varsigma^\mu \text{idcyc+1} \mod l_\mu \varsigma^\mu \text{idcyc} \\
\text{wis} & \mapsto \text{wis} + \varepsilon_b \varsigma^\mu \text{idcyc+1} \mod l_\mu \chi^\mu 
\end{align*}
\]

4. The two learning noise parameters \( l_n \) and \( l_{nb} \) are initialized with predefined values.

5. Following iterations are performed \( n_{test} \) steps, for \( \text{idcyc} \) going from 1 to \( l_\mu \):
   
   (a) Using \( l_{nb} \), a noisy version \( \chi^\mu_{lnb} \) is generated as an external stimulus;
   
   (b) Using \( l_n \), a noisy version \( \varsigma^\mu \text{idcyc} \) is generated as initial internal states;
   
   (c) To skip the transient, a multiple of \( l_\mu \) synchronous simulations is performed;
   
   (d) The internal state is stored: \( x_{current} \)
   
   one more simulation is performed: \( x_{current} \mapsto x_{next} \).
   
   (e) For each neuron \( i \), the equality \( \text{sign}(x_{next,i}) \overset{?}{=} \varsigma^\mu \text{idcyc+1} \mod l_\mu \) is tested.

If they are not equal,

- all the synaptic connections impinging on this neuron are modified according to the Hebbian learning rule:

\[
\begin{align*}
\text{wij} & \mapsto \text{wij} + \varepsilon \varsigma^\mu \text{idcyc+1} \mod l_\mu x_{current,j} \\
\text{wis} & \mapsto \text{wis} + \varepsilon_b \varsigma^\mu \text{idcyc+1} \mod l_\mu \chi_{lnb,i} 
\end{align*}
\]

- we go back to step “1” and restart with the whole data set.

\( \Rightarrow \) Once here, data \( \mu \) is stable. The next steps aim at improving robustness to noise.

**learning with noise**

⇒ once here, data \( \mu \) is stable. The next steps aim at improving robustness to noise.

\( \Rightarrow \) Once here, the data \( D^\mu \) corresponds to a stable and robust attractor of the internal dynamics. \( D^\mu \) is removed from \( D \). The whole algorithm is repeated from step “2”.
Two noise parameters are used to test the ability to recover from noise during the retrieval phase: \( n_u \), describing the noise injected in one of the learned cycle’s patterns, \( \Sigma_{n_u idcyc} \), and \( n_u b \) describing the noise injected in the external stimuli, \( \xi_{n_u b} \).

After injecting some noise, specified by the couple \((n_u, n_u b)\), 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). Two kinds of measures have been used: the Hamming distance between the two patterns (Equation 6.10) and the overlap \( m^\mu \) between them given by:

\[
 m^\mu = \frac{1}{N} \sum_{i=1}^{N} \frac{\Sigma_{i idcyc}^\mu}{\Sigma_{i noisy}^\mu}
\]

thus, if the two patterns match perfectly, this overlap equals to 1, while it equals to \(-1\) in the opposite case. The overlap \( m \) and the Hamming distance \( d_h \) are related by:

\[
 d_h = N \frac{m + 1}{2} \quad \text{where } N \text{ is the number of neurons}
\]

The reason for using not only the Hamming distance but also the overlaps is to enable direct comparison with the results from (Forrest and Wallace, 1995), where the fraction of properly recalled nominal states is plotted as a function of increasing initial overlaps \( m_0^\mu \).

### 3.1 Results on auto-associative memories

Results obtained from the Hebbian algorithm when storing patterns in an auto-associative way are presented in Figure 6.2. First, the data set described in Equation 6.4 (without external stimuli) is tested (dotted curves). Then, enhancements obtained by using external stimuli (plain curve) are quantified by testing the data set described in Equation 6.5.

This figure shows the rate of properly recalled patterns as a function of the noise imposed on the initial pattern (quantified by the initial overlap \( m_0 \)). For the second type of tested data set, this pattern not only initializes the initial state, but also continuously feeds the external stimulus.

From the dotted curves, it clearly appears how training with noise enhances content addressability. These results are in line with the results presented in (Forrest and Wallace, 1995).

The maximum of content addressability is obtained by the addition of external stimuli (plain curve). Since the same noisy pattern feeds the external stimulus and initializes the internal state, \( \xi_{n_u b}^\mu = \xi_{n_u}^\mu \) (or \( n_u = n_u b \)), this curve can be straightforwardly compared to the other curves obtained without external stimuli. For instance, for an initial overlap \( m_0 = 0.4 \), the use of external stimuli increases the number of patterns properly recalled by \( \sim 20\% \).
3.2 Results on hetero-associative memories

The precedent section shows that the content addressability of the auto-associative memories is enhanced by duplicating the information used to store and to retrieve patterns in an Hopfield network. The same information is used as an external stimulus and to initialize the network’s internal states.

In this subsection, the external stimulus is chosen independently from the stored pattern. Thus, each stored data is defined by a pair of patterns (defined in Equation 6.6). As explained in Section 2.4.2, the addition of external stimuli increases the number of parameters to be tuned ($ln$, $ln_b$, $\varepsilon$, $\varepsilon_b$). There is no perfect way of choosing between these parameters: they need to be adjusted in function of a given problem which is defined by the expected values of $nu$ and $nu_b$.

The 3 plots appearing in Figure 6.3 show how noise robustness, represented by the Hamming distance between the stored pattern and the recalled pattern, varies in function of both the noise injected in the external stimuli and in the initial states. It has to be noted that for these experiments, initial overlaps go from $-1$ to $1$ (and not $0$ to $1$). Each plot represents statistics obtained from one hundred networks (of 100 neurons) which had to learn 25 randomly chosen patterns ($\alpha = 0.25$).

The first figure, provided as a reference, tests a case without external stimuli (all weights are set to zero: $w_{ij} = 0$). As an obvious consequence, no impact is observed when external stimuli are modified through $m_{0b}$. For example, this figure shows that...
a pattern having 25% of noise (i.e. \( m_{0s} = 0.5 \)), is perfectly recovered all the time.

The two following figures show the importance of the learning parameters’ tuning. In the last figure, data are learned with no noise on external stimuli (\( ln_b = 0 \)) while noise is injected on the pattern for storage in the attractor (\( ln = 15 \)). Since external stimuli are never in error (\( ln_b = 0 \)), great confidence is given to them. As a result, when testing robustness to noise, if the external stimulus is perfectly given (\( m_{0b} = 1 \)), the stored pattern is almost all the time recovered: in the worst case (when initial states are randomly chosen, i.e. \( m_{0s} \approx 0 \)), in average only 15% of error is obtained in the recovered attractor. However, if external stimuli are noisy the performance decreases fast. The middle figure shows an intermediate case where noise is injected on the external stimuli during the learning phase (\( ln_b = 6\% \)). As a consequence, less confidence is given to them and after learning the network is able to recover internal patterns from noisy stimuli.

![Figures](image1.png)

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

Figure 6.3: These plots show the content addressability obtained for the learned networks: the Hamming distance between the expected pattern and the obtained one is plotted in function of the noise injected both in the initial states \( (m_{0s}) \) and in the external stimulus \( (m_{0b}) \). Each image has been obtained for different values of the noise parameters during learning. The left image works without stimuli. \( N = 100 \) and \( \alpha = 0.25 \).

As a matter of fact, this fully recurrent network is used here for an associative task which could be achieved by a classical and static feedforward network: for a given external stimulus (the input layer), we want an associated pattern in the internal state (the output layer). However, differences exist and justify the absence of comparisons. In a feedforward network, one given input is associated with a specific output. Here, the obtained output is a function not only of the stimulus but also of the initial internal state: a same stimulus can lead the network to converge to different attractors depending on its initial state. Figure 6.4 shows such an example of a hetero-associative data set where two identical external stimuli lead to different internal states with regard to their initial values.
Learning using an iterative supervised Hebbian algorithm

Figure 6.4: The left part of the figure shows an example of an hetero-associative data set composed of four data \(((1, A), (1, \alpha), (2, B), (2, \beta))\). The right part shows the evolution of the internal state after having been initialized with the corresponding external stimulus and with a noisy internal state pattern.

3.3 Learning Sequences of patterns

In this subsection the network learns sequences of patterns, each of them being indexed with an external stimulus.

Figure 6.5 shows the network’s capacity in the absence of noise during the training. By defining \( q_T \) as the total number of patterns (the sum of all the patterns in all the sequences) we can see that the load parameter \( \alpha_T = q_T/N \) remains more or less constant \( \alpha_T \approx 1.5 \). For example, it is possible to store 76 sequences of size 2 or 14 sequences of size 10 in a 100 neurons’ network (showing that the storing capacity is independent from the size of the sequences to learn). The same figure shows that using the external stimuli slightly increases the storing capacity (82 sequences of size 2, 16 sequences of size 10 for example, \( \alpha_T \approx 1.6 \)).

The presence of external stimuli has another impact: it enhances the content addressability of the learned sequences. Figure 6.6 shows the results obtained with different learning set parameters.

In the absence of noise in the external stimuli, great confidence is given to them (as shown in Figure 6.6(c)): if the exact pattern is settled in the stimulus, even when the internal state is completely random, the probability of retrieving the learned sequence is still very high \( \approx 80\% \). However, if noise is slightly added onto the stimulus, the learned sequence is lost. When the noise parameter is different from zero \( \ln b = 6\% \), then it is possible to converge to the learned sequence, even in the presence of noise in the stimulus. These curves can be compared to the curve obtained without using the stimuli.

The existence of the external stimuli enables the storage of sequences sharing some of their patterns, as appearing in the data set given Figure (6.7). For example, without external stimuli it would be impossible to store the two sequences “colin” and “chaos” in the same network. In the first sequence the letter “C” is followed by an “O” while in the second one, it is followed by an “H”. Only the presence of external stimuli allows
3 Experiments and Results

Figure 6.5: Storing capacity of a recurrent network. This plot shows the number of iterations needed to store sequences of patterns as a function of the number of learned sequences. Computations have been performed for sequences of different size (respectively: 2, 4, 6 and 10, from right to left). The plain curves give the performance of storing with external stimuli while dotted ones show the performance without. External stimuli slightly enhance the maximum capacity. N=100

the distinction between the two cases, both during the learning and the retrieving phases.

3.4 Learning incrementally: relying on the iterative process

In contrast with the Hebbian prescription rule used by Hopfield (Equation 6.8), the learning procedure defined here is no longer a one time step process: it defines an iterative algorithm where weights are adjusted incrementally. As a consequence, after the learning of a given data set, it is possible to add new data to learn and to continue the learning with this data without loosing all previous information.

By using this property, interesting results could be obtained in practical applications. For example, suppose that one data has been learned in a network’s limit cycle attractor in a given amount of time steps. Then, suppose that, without resetting the network’s weights, another data is learnt by the same network. After this second learning, it is highly probable for the first data to be lost. But with some chance, nothing is lost: the previously learned limit cycle attractor could have been moved only slightly, and if we decide to learn the first data again, it is highly probable that the learning time will be greatly reduced. However, now the second data could be lost. But, by looking at the obtained attractor, we can nearly see it. By repeating this
Learning using an iterative supervised Hebbian algorithm

Figure 6.6: After learning 5 data of size 5 cycles, the Hamming distance between the expected sequences of patterns and the one obtained is plotted in function of the noise injected both in the initial states ($m_0s$) and in the external stimulus ($m_0b$). Each image has been obtained for different values of the noise parameters during learning. The left image gives results obtained without stimuli. $N = 100$ and $\alpha = 0.25$.

Figure 6.7: Example of a data set made of sequences sharing letters. The 4 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.

4 Conclusion

This work presents a Hebbian learning algorithm for storing fixed patterns and cycles of patterns in the fixed point and limit cycles attractors of a Hopfield network.

The basic version of the implemented algorithm is nothing new. However, for this basic and classical case no results exist with regard to the storing capacity and content addressability when learning cycles of patterns.
A modification of the learning and retrieving algorithm has been proposed. It consists of using external stimuli to enhance performances. 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 parallel with this enlargement, both a slight increase of the storing capacity and an improvement of the robustness to noise have been observed.

The addition of these external stimuli enables comparison with the classical feedforward perceptron approach. In both cases, we have two layers: the input layer is given by the external stimulus and the output layer is represented by the “internal state”. However, basic differences remain. First of all, in the recurrent network, a given stimulus can lead to two different attractors depending on the initial internal state. Secondly, the absence of recurrence in a classical feedforward network prevents the storing of cycles. Thirdly, stimuli enable the storage of sequences sharing common patterns.

This chapter has shown different quantitative results. For example, for some values of the learning noise parameters, when learning 20 patterns of size 2 in a 100-neurons RNN, the stimulus enables the recovery from a fully random internal state with more than 20% of probability. The next chapter provides a closer look to what is happening in the other 80% of cases, and therefore focuses on dynamical considerations.
Chapter 7

Dynamical observations on networks learned through an iterative supervised Hebbian algorithm

1 Introduction

The last chapter has focused on the implementation of an Hebbian based learning algorithm fulfilling the requirements of our learning task: learning external stimuli in limit cycle attractors of the RNN’s inner dynamics. Numerical results have been computed assessing both the encoding capacities and the tolerance to noise of this algorithm.

The goal of this chapter is to study the underlying dynamics of a network after learning by means of an Hebbian algorithm. The algorithm used has been already described in detail in the previous chapter. However, to enable the use of dynamical tools such as the computation of Lyapunov exponents and power spectrum analysis, the algorithm is adapted here to continuous state neurons.

Considerable differences will appear depending on the learning task: if the storage of static patterns stabilizes the network, the coding of information in robust cyclic attractors increases the network’s chaos, and the more chaos appears as a regime in the back, erratically itinerating among brief appearances of these attractors.

The next section describes both the modification required on the model by using continuous neurons and the consequent modifications necessitated on the learning algorithm. Section 3 summarizes the results obtained.
2 Modifications entailed by the use of continuous neurons

Compared to the previous chapter, a major modification of the model is performed here. To enable the use of the dynamical tools developed, the neurons’ discrete activation function has been turned into a continuous activation function.

2.1 Model’s modification: use of a sigmoid-like activation function

The \( \tanh \) function is chosen, hence internal neuron states range from \(-1\) to \(1\). To be able to compare binary patterns with internal states of the continuous network, a filter layer is added based on the sign function:

\[
\begin{align*}
  x_i < 0 & \mapsto o_i = -1 \\
  x_i \geq 0 & \mapsto o_i = 1 
\end{align*}
\]  

(7.1)

where \( x_i \) is the internal state of the neuron \( i \) and \( o_i \) is its associated output (i.e. its visible value).

In fact, we are already familiar with this filter layer. The same mechanism of symbolic quantization of the output space, introduced in previous chapters, has shown how these networks can be exploited to encode information through cycles of the underlying dynamical system. Here the quantization level equals 2.

Figure 7.1 represents a sequence of size 2 unfolding in a network of four neurons. The 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. This figure is similar to figure 6.1 with the addition of the filter layer.

2.2 Adaptation of the algorithm to continuous activation functions

In the previous chapter, the iterative supervised Hebbian algorithm has been developed for neurons having discrete activation functions. As a consequence, the pseudo code of the algorithm described in Table 6.1 is limited to neurons with discrete activation functions. When working with continuous state neurons, this algorithm needs to be adapted in order to prevent the learned data from vanishing after a few iterations. This adaptation consists of waiting 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:

\[
\forall \mu, \nu \text{ if } (x(0) = s^{\mu,\nu}) \mapsto (x(1) = s^{\mu,\nu+1}) \Rightarrow \text{stop}
\]  

(7.2)
Figure 7.1: Picture of a fully recurrent neural network fed by an external stimulus (which appears in the part (a)). Three shots of the network are shown (b). Each one of them represents the internal state of the network at a successive time step. After filtering, a cycle of size 2 can be seen (c).

while it becomes for continuous neurons:

$$\forall \mu, \nu \exists f \left( x(0) = \xi^{\mu,\nu} \right) \Rightarrow \left( o(1) = o(l_\mu + 1) = \ldots = o(T * l_\mu + 1) = \xi^{\mu,\nu+1} \right) \Rightarrow \text{stop}$$

(7.3)

where $T$ is a further parameter of our algorithm (set to 10 in all the experiments).

The Figure 7.2 shows how a learned data vanishes when such a test is not performed.

Figure 7.2: This figure respectively shows the internal state and the filtered output of a 10x10 network, at successive time steps. The cycle “KO” has been learnt 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 and at the end of the second cycle, differences appear in the output. The learned sequence vanishes progressively.
3 Experiments and Results

Quantitative results with regard to the efficiency of the implemented algorithm in terms of its storing capacity and the content addressability of the stored patterns have been presented in the previous chapter. The experiments described were restricted to networks of discrete state neurons. Similar quantitative results have been obtained for continuous neurons, allowing the direct focus on the impact of iterative supervised Hebbian learning on the underlying dynamics of fully connected networks.

Learning tasks leading to networks having very different behaviors have been compared, i.e. the learning of static patterns, the learning of sequences of patterns and the learning of one sequence of increasing size. Quantitative analyses of the dynamics encountered in learned networks have been performed using two kinds of measures: the mean Lyapunov exponent and the probability to have chaotic dynamics. Both measures come from statistics on a huge number of learned networks (here 1000 networks have been analyzed). For each learned network, dynamics obtained by randomly varying the external stimuli and the initial states have been tested (1000 different configurations). These tests aimed at analyzing the so-called background or spontaneous dynamics obtained by stimulating the network with external stimuli and initial states different from the learned ones.

Each time the Lyapunov exponent was computed. 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 (in practice: the dynamics is considered as chaotic if the Lyapunov exponent is greater than 0.01).

The obtained results are made more meaningful by comparing global dynamics of learned networks with global dynamics of networks obtained randomly (without learning). However, to constrain random networks to behave as a kind of surrogate network, they must have the same means $\mu$ and the same standard deviation $\sigma$ for their weight distributions (neuron-neuron and stimulus-neuron):

$$\begin{align*}
\mu(w_{ij}^L) &= \mu(w_{ij}^C) & \mu(w_{bi}^L) &= \mu(w_{bi}^C) & \mu(w_{ii}^L) &= \mu(w_{ii}^C) \\
\sigma(w_{ij}^L) &= \sigma(w_{ij}^C) & \sigma(w_{bi}^L) &= \sigma(w_{bi}^C) & \sigma(w_{ii}^L) &= \sigma(w_{ii}^C)
\end{align*}$$

These “constrained random networks” enable the measurement of the weight distribution’s impact. Chapter 4 has shown how the results heavily depend on the shape of the weight distribution (the uniform and the gaussian random distributions were tested). In order to choose the most pertinent random distribution, the next section plots the weight distributions of learned networks.

3.1 Weight distributions after learning

Figures 7.3 and 7.4 show the weight distributions of learned networks when the learning task respectively is the storing of static patterns and the storing of one cycle of increasing size. Moreover, in both cases, the impact is shown when making the respective learning tasks more complex.
The first observation is that all these curves look gaussian. In the next sections we will compare the background dynamics obtained for learned networks with the background dynamics obtained for random networks having a similar, but random, weight distribution\(^1\).

The second observation is that as the learning tasks are strengthened, weight distributions are widened. Similar trends have been observed when studying the impact of the BPTT learning.

The third observation is that on average, learning one huge cycle leads to slightly negative auto-connections, whereas learning static patterns leads to the opposite trend, i.e. auto-connections become more and more positive.

![Graphs showing weight distribution](image)

(a) Weight distribution after Hebbian learning of 4 static data

(b) Weight distribution after Hebbian learning of 50 static data

Figure 7.3: Weight distribution after Hebbian learning of static patterns. \(N = 100, \ln_s = 6\% \text{ and } \ln_b = 6\%\)

### 3.2 Network stabilization through Hebbian learning of static patterns

Figure 7.5 shows the mean Lyapunov exponents and probabilities to have chaos for networks with learned data encoded in fixed point attractors only. To avoid weight distribution to converge to the identity matrix (if \(\forall i, j : w_{ii} > 0 \text{ and } w_{ij} \approx 0\), all patterns are obviously learned but without robustness), noise has been added during the training period. We can observe that:

\(^1\text{however, similar results have been obtained using a uniform distribution}\)
Dynamical observations on learned networks

Figure 7.4: Weight distribution after Hebbian learning one cycle. \( N = 100, \ln s = 0\% \) and \( \ln b = 0\% \)

- mean Lyapunov exponents of learned networks are always negative: by making the learning task more complex, networks are more and more constrained - the mean Lyapunov exponent increases but still remains below a negative upper bound;

- it is nearly impossible to find chaotic dynamics for spontaneous regimes in learned networks, even after intensive learning of static patterns;

- random networks having same global properties show very different results, clearly indicating how the learned weight distribution is anything but random.

Hopfield has proven that RNNs having no auto-connections and symmetric weights necessarily converge to fixed points attractors while asynchronously updated. This has been demonstrated by identifying the energy (or Lyapunov) function for this system:

\[
H = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} x_i x_j
\]  

(7.5)

Each state variation produced by the system’s equation entails a non-positive variation of \( H: \Delta H \leq 0 \). Let us demonstrate this assertion. Suppose that at time \( t \) the potential impinging neuron \( k \) is such that the neuron’s state is flipped. The energy
3 Experiments and Results

Figure 7.5: Mean Lyapunov exponents and probability of chaos in RNNs after Hebbian learning of static patterns. Mean and standard deviation are plotted. These curves are compared with results obtained from RNNs with randomly chosen weights from a gaussian distribution having the same global properties. The corresponding curves are above the previous ones. Network stabilization through Hebbian learning of static patterns can be observed. $N = 100$, $ln_s = ln_b = 6\%$

The function at time $t$ is given by:

$$H(t) = -\frac{1}{2} \left[ \sum_{i \neq k} \sum_{j \neq k} w_{ij} x_i(t) x_j(t) + x_k(t) \sum_{i \neq k} (w_{ik} + w_{ki}) x_i(t) + w_{kk} x_k(t) x_k(t) \right]$$

(7.6)

After the flip of neuron $k$, since $\forall i \neq k: x_i(t + 1) = x_i(t)$, the difference of energy is given by:

$$\Delta H_k = \frac{1}{2} \left[ \Delta x_k \sum_{i \neq k} (w_{ik} + w_{ki}) x_i(t) \right]$$

$$= \frac{1}{2} \left[ 2 \Delta x_k \sum_i w_{ki} x_i(t) - 2 \Delta x_k w_{kk} x_k(t) + \Delta x_k \sum_{i \neq k} (w_{ki} - w_{ik}) x_i(t) \right]$$

(7.7)

where $\Delta x_k = x_k(t + 1) - x_k(t)$. The first term in the last equation is always positive since $\sum_i w_{ki} x_i(t)$ is the potential impinging the neuron $k$ at time $t$ and hence it has the same sign as $\Delta x_k$. The sign of the second term is given by $w_{kk}$ since $x_k(t) \Delta x_k$ is always negative. The third term is more risky to predict.

To ensure always decreasing energy and reaching a given minimum value (the fixed point attractors), Hopfield’s conditions are very straightforward: null auto-connections (to have the second term equal to null) and symmetric weights (to have the third term equal to null).

In our case, the iterative Hebbian learning algorithm described here prevents the assertion of a decreasing law for the energy function. However, since we are still basing ourselves on a symmetric Hebbian rule, it is likely to have mostly symmetric connections (this is verified experimentally). Therefore, the third term cannot become too
negative. Moreover, auto-connections tend to become positive, ensuring the second term to be positive and relativizing the impact of the third term. As a consequence, the function $H$ nearly all the time remains constant or decreases, and hence our system remains most of the time similar to a dissipative system stabilizing to energy minima.

When working with random networks built with global properties obtained from learned networks, the second term (the auto-connections) stays positive of course. However, connections between neurons are now very unlikely to be symmetric. Hence the third term may become huge and the energy is no longer decreasing all the time. This explains these networks’ tendency to become unstable, as appearing in Figure 7.5(b).

When learning cycles, the Hebbian rule and thus the weights are no longer symmetric, there are no longer guarantees for convergence to fixed point attractors. Since we do not want fixed point attractors but limit cycles attractors, we hope that convergence does not happen.

### 3.3 Hebbian learning of cycles: a new road to chaos

If learning data in fixed point attractors stabilizes the network, learning sequences in limit cycle attractors leads to diametrically opposed results. Figure (7.6) shows the global dynamics obtained after learning an increasing number of sequences of different size. Figure 7.7 provides the results obtained after storing one sequence of increasing size in a limit cycle attractor of the network. At this stage, similar results are obtained when storing one cycle or many cycles. We can observe that:

- by increasing the learning data set, spontaneous dynamics of learned networks become more and more chaotic. As a matter of fact, chaos prevails as the spontaneous or background dynamics;

- in comparison with constrained random networks, learning contributes to structuring the dynamics (at a minimum the network has to remember the learned data!);

- characteristics of learned networks appear to fall outside the range of values obtained for random networks. However, by intensifying the learning task, the differences between the random and the learned networks tend to vanish.

Obtaining chaos by learning an increasing number of cycles using an Hebbian mechanism can be seen as a new road to chaos. Such roads to chaos are illustrated in Figure 7.8. 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.

In contrast with the classical roads shaped by the gradual variation of a control parameter, this new road relies on an external mechanism simultaneously modifying a set of parameters to fulfill an encoding task.
Figure 7.6: Dynamical properties of networks after Hebbian learning of multiple sequences of patterns, respectively of size 2, 4 and 10. Mean and standard deviation are plotted. These curves are compared with the results obtained from RNNs with randomly chosen weights from a gaussian distribution having the same global properties. The corresponding curves are above the previous ones. As the data set increases in complexity, the dynamics become fully chaotic. \( N = 100, \ln s = \ln b = 0 \).

Figure 7.7: Mean Lyapunov exponents and probability of chaos in RNNs after Hebbian learning of one cycle of increasing size. Mean and standard deviation are plotted. These curves are compared with the results obtained from RNNs with randomly chosen weights from a gaussian distribution having the same global properties. The corresponding curves are above the previous ones. As the cycle’s size increases the dynamics become fully chaotic. \( N = 100, \ln s = \ln b = 0 \).
Dynamical observations on learned networks

(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 7.8: Road to chaos observed in recurrent networks when learned through an iterative supervised Hebbian algorithm (the number of learning iterations is indicated on the x-axis). The network is initialized randomly and is following a given transient, the state of one neuron chosen randomly is plotted on the y-axis.
3.4 Characterization of the chaotic dynamics

While the precedent section has given a quantitative analysis of the “chaoticity” of the learned networks, this section aims at introducing a more qualitative description of the different types of chaotic regimes encountered in these networks.

For this purpose, return maps and power spectra analysis turned out to be very useful. Three types of chaotic regimes have been identified in these networks. Figure (7.9) shows the power spectra and the return maps of these three generic types. For each of them, a power spectrum has been plotted both for one particular neuron and for the network’s mean signal. We can observe that:

**white noise** The power spectrum of this type of chaos (Figure 7.9(a)) shows a total lack of structure, making it very similar to white noise (all the frequencies are represented nearly equally). The associated return map is completely filled, with a bigger density of points at the edges, indicating the presence of saturation. No useful information can be obtained from such chaos;

**deep chaos** The power spectrum of this type of chaos shows more structure (Figure 7.9(b)). The chaos obtained is very deep and not so far from white noise. However, broad peaks and their harmonics are clearly distinguishable. The associated return map is very similar to the previous one and does not seem to provide any useful information;

**informative chaos** The power spectrum of the last type of chaos looks the most informative one (Figure 7.9(c)). Peaks show up (two in this case), usually with one of them clearly pronounced indicating the presence of a nearby limit cycle. The other periods indicate unpredictable chaotic itinerancy. The associated return map shows more structure. In fact, by slightly modifying the external stimuli, the network falls in a nearby limit cycle attractor.

It is almost possible to predict the type of chaos preferentially encountered as a function of the complexity and the size of the data set to be learned.

For instance, when learning one cycle of increasing length, the dynamics goes through a deep chaos (Figure 7.9(a)) to a white noise one. When learning a small set of cycles, the chaotic dynamics is preferentially of the third type (the informative chaos). The dynamical structure reveals the existence of nearby competing attractors. Figure 7.9(c) shows the phenomenon of frustration obtained when the network hesitates between two (or more) nearby cycles, passing from one to another. These figures have been obtained by setting the external stimuli randomly switching in-between learned ones. By increasing the set, the dynamics looses their structure and tends to behave as white noise. All information about hypothetic nearby limit cycles attractors is lost.

Chaotic dynamics observed in “constrained random networks” are generally similar to white noise. This could explain the large mean Lyapunov exponent obtained for these networks. Figures 7.9(a) have been obtained for such networks.

All these observations lead us to propose the classification summarized in Table 7.1.
(a) Chaotic regime obtained
(b) Chaotic regime obtained in a network which has learned learning 5 size 10 cycles. The lack of structure makes it very similar to white noise.
(c) Chaotic regime in after learning 5 size 10 cycles. The different peaks show the presence of nearby limit cycles among which the network hesitates.

Figure 7.9: When presenting in-between stimuli, very characteristic power spectra and return maps are plotted for chaotic regimes encountered in random and learned networks. Each time, the figures at the top show return maps of the mean signal, the center figures show return maps of a particular neuron and the figures below show power spectra obtained from the mean signal. N=100
Table 7.1: Classification of chaotic dynamics occurring in RNNs in function of the procedure followed to build the weights matrix.

<table>
<thead>
<tr>
<th></th>
<th>Chaos ~ white noise</th>
<th>Deep chaos</th>
<th>“informative” chaos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Networks</td>
<td>A</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>H.L. at saturation</td>
<td>A</td>
<td>S</td>
<td>N</td>
</tr>
<tr>
<td>H.L. one small cycle</td>
<td>N</td>
<td>A</td>
<td>S</td>
</tr>
<tr>
<td>H.L. few cycles</td>
<td>N</td>
<td>S</td>
<td>A</td>
</tr>
</tbody>
</table>

where H.L. means Hebbian Learning, “A” means most of the time, “S” means sometimes and “N” means almost never;

### 4 Conclusion

The last chapter described the implementation of a supervised iterative Hebbian learning algorithm which has proven to give good results in terms of encoding capacity. It was shown that when using external stimuli to index the stored data, the entire internal dynamics change, leading to an enlargement of the set of attractors and potential “memory bags”.

The different spontaneous dynamics occurring in these learned networks have been reviewed in this chapter. These dynamics have been further compared with the global dynamics of networks obtained randomly (without learning) but constrained by the same global parameters as the learned networks they were compared to (same mean and same variance).

It has been shown, and theoretically validated, 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. Similarly, different types of chaos have been observed: from a “frustrated” chaos showing strong presence of nearby limit cycle attractors to a very deep chaos nearly similar to white noise.

It appeared that the chaotic dynamics occurring in constrained random networks is most of the time similar to white noise. For learned networks, the probability to observe a given type of chaos is highly related to the amount and to the type of the data to learn. At the limit of their capacity, the chaotic dynamics observed in RNNs are often similar to white noise. By diminishing the amount of learned data, chaotic dynamics shows more structure. In networks which learn a few number of sequences, a very characteristic and informative type of chaos can be observed, i.e. the frustrated chaos. The observation of their strange attractors show the strong presence of nearby limit cycle attractors: by tuning the stimulus in a configuration in-between two learned stimulus leading to different stable attractors, the network hesitates from one limit cycle to the other in an unpredictable way.
Chapter 8

Learning using an iterative unsupervised Hebbian algorithm

1 Introduction

An iterative supervised Hebbian learning algorithm has been introduced in chapter 6 to store a given data set on a fully connected recurrent neural network (RNN). Every information stored in the network is made of a pair of data: the stimulus and the corresponding limit cycle attractor. After learning, if the network receives a learned stimulus (or a noisy version of this stimulus), and if its initial internal state is not too far from the expected limit cycle attractor (or at least from one of its patterns), the network’s dynamics is expected to converge to this limit cycle attractor. Using external stimuli has proven to enhance both the storing capacity and the content addressability of the learned sequences.

Results obtained from this algorithm are far better than the results obtained from the global gradient based algorithm (see Chapter 5). However, they remain far below the capacities obtained in random networks: learning seems to be too constrained.

In order to get a better understanding of the learning impact on the networks’ underlying dynamics, this algorithm has been adapted to continuous neurons in the last chapter. This has enabled us to use dynamical tools such as the computation of Lyapunov exponents and power spectrum analysis on the learned networks. This has shown how the learning of sequences in limit cycle attractors tends to increase the dimension of the potential attractors. The more you learn using limit cycles attractors, the more chaotic dynamics is observed. In the end the network is no longer able to manage anything and it turns out to be fully and strongly chaotic.

The aim of this chapter is to try to relax the learning task in order to avoid this road to chaos and to have better results in terms of the encoding capacity. In chapter 4, no learning task was specified: the performance of random networks was quantified by computing the number of potential robust attractors with a brute exhaustive search. In chapters 5, 6 and 7, two different learning algorithms have been tried, i.e. one being global (the gradient based BPTT algorithm), the other local (an iterative adaptation...
of a Hebbian algorithm). Both have been supervised: when being in a specific initial state and when fed by a specific external stimulus, the network was expected to converge to a predefined limit cycle attractor. From a biological point of view, such a way of learning does not really make sense. Actually it is very unlikely that the cycles occurring in the brain when performing cognitive tasks have a predefined meaning.

This chapter introduces an unsupervised learning task, which is more plausible from this biological point of view: the network has to learn to react to an external stimulus by cycling through a sequence which is not specified a priori. This perspective remains in line with a very old philosophical conviction called constructivism and was modernized in neural net terms by several authors (among others (Varela et al., 1991), (Erdi, 1996), (Tsuda, 2001)). In this view, the human brain is hermeneutic (interpretative) by nature: it does not only perceive but also creates new realities. One operational form has achieved great popularity as a neural net implementation of statistical clustering algorithms ((Kohonen, 1982), (Grossberg, 1992)). The algorithm to be presented now can be seen as a dynamical extension of these preliminary works where the coding scheme relies on cycles instead of on single neurons.

An unsupervised version of the iterative Hebbian algorithm has been developed in this chapter. By comparing this algorithm with the supervised one, promising results have been obtained: this less constrained algorithm greatly enhances both the storing capacity and the content addressability of the learned networks as well as the computational time. Furthermore, convincing results were achieved when comparing the dynamics of learned networks. This time, less chaotic dynamics showed up and in general chaotic dynamics were more structured, i.e. they were made from brief itinerancy among learned cycles, and thus could be related to the “frustrated chaos” described by Bersini (Bersini, 1998), or to the chaotic itinerancy ((Ikeda et al., 1989), (Kaneko, 1992), (Tsuda, 1992)).

Since the model is identical to the one described in the previous chapter - a fully connected RNN using continuous activation neurons and discrete time step - we can immediately focus on the implementation of the unsupervised algorithm. Different results will be analyzed.

2 The unsupervised Hebbian learning algorithm

2.1 Description of the unsupervised learning task

This new algorithm is characterized by the fact that the information is not fully specified a priori: only the external stimuli are known before learning.

The limit cycle attractor associated with an external stimulus is identified through the learning procedure. 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 learning, the data set is defined by $D_{bl}$ (bl standing for “before learning”):

$$D_{bl} = \{ D_{bl}^1, \ldots, D_{bl}^q \}$$

where $D_{bl}^\mu$ is the external stimulus feeding the network. After learning, the data set becomes:

$$D_{al} = \{ D_{al}^1, \ldots, D_{al}^q \}$$

where where $\zeta^{\mu,i}$ is the pattern $i$ of the cycle to learn. $l_{\mu}$ is the size of this learned cycle.

### 2.2 The iterative supervised Hebbian algorithm

Before focussing on the implemented unsupervised Hebbian algorithm, this subsection first summarizes the iterative supervised Hebbian learning algorithm introduced in chapter 6. Two reasons justify this digression. First, the implemented unsupervised algorithm is built on the top of the supervised algorithm. Furthermore, it will appear that a supervised learning algorithm is still needed to retain all the information previously learned.

The discrete version of the supervised algorithm was inspired by (Forrest and Wallace, 1995) and (Hertz et al., 1991). The principle can be described as follows: at each learning iteration, the stability of every nominal pattern $\xi^\mu$, 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:

$$w_{ij} \mapsto w_{ij} + \varepsilon_s \zeta^{\mu,i} \xi_j^\mu$$

$$w_{is} \mapsto w_{is} + \varepsilon_b \chi^\mu \xi_i^\mu$$

**static case**

$$w_{ij} \mapsto w_{ij} + \varepsilon_s \zeta^{\mu,v+1} \zeta_j^{\mu,v}$$

$$w_{is} \mapsto w_{is} + \varepsilon_b \chi^\mu \chi_i^\mu$$

**cyclic case**

where $\varepsilon_s$ and $\varepsilon_b$ respectively define the learning rate and the stimulus learning rate.

This algorithm has been improved by adding explicit noise during the learning phase. In fact, although the precedent procedure guarantees a perfect storage, nothing guarantees the robustness of the attractors. A slight modification of the internal state of one neuron could result in the network bifurcating into another attractor. In order to not only store the patterns, but also to ensure a sufficient enough content-addressability, we had to try to “excavate” the basins of attraction. For this purpose, domains of attraction around each nominal pattern were enlarged by training the network to associate noisy versions of each nominal pattern with the desired pattern, following a given number of iterations expected to be sufficient for convergence.

In the last chapter this algorithm has been adapted to continuous state activations neurons. A filter layer was added to quantize the output. This allowed comparison with the desired bit-pattern and enabled symbolic investigations on the dynamical attractors.
2.3 The iterative unsupervised Hebbian algorithm

The main difference between the unsupervised Hebbian algorithm and the supervised one lies in the nature of the information learned. In the supervised version, each information to learn is given a priori and is fully specified. In the unsupervised 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 $D_{bl}$ to learn (Equation 8.1);
- a range $[\min_{cs}, \max_{cs}]$ which defines the bounds of the accepted periods of the limit cycle attractors coding the information;

The unsupervised learning algorithm can be broken down in three phases which are constantly iterated until convergence:

**stimulation of the network** During this phase, the network only receives the external stimulus, consequently trapping it in an attractor output $\mu$;

**proposal of an attractor code** Since the idea is to constrain the network as little as possible, the meaning is assigned to the stimulus by associating it with a new version of the attractor output $\mu$, called cycle $\mu$, respecting the periodic bounds $[\min_{cs}, \max_{cs}]$ and being “original”;

**learning the information** Once the new attractor cycle $\mu$ has been proposed, it will be tentatively learned by relying on a supervised procedure. A limited number of the iterations of the supervised algorithm is performed in order to avoid constraining the network too much.

The pseudo-code of this unsupervised learning algorithm is described in Table 8.1.

This learning algorithm is incremental since the learning of new information can be done by preserving all information that has already been learned.

**Enforcing robustness through trials-errors-adaptations**

Since this algorithm is based on a supervised algorithm, it is still possible to rely on the parameters $ln$ and $ln_b$ described in Chapter 6, Section 2.3.1. These two learning noise parameters represent the noise injected during the learning phase, respectively on the internal states and on the stimulus. This allowed the explicit excavation of the learned attractors’ basins of attractions.

---

1 original means that each pattern composing the limit cycle attractor must be different from all other patterns
Table 8.1: Pseudo code of the iterative supervised Hebbian algorithm

stimulation of the network

1. \( \forall \) data \( D_{\mu}^b \) to learn
   
   (a) the stimulus is initialized with \( \chi^\mu \);  
   (b) the states are initialized with \( \varsigma^{\mu,1} \) which are obtained from the previous iteration (or random at first);  
   (c) to skip the transient, the network is simulated some steps;  
   (d) the states \( \varsigma^{\mu,i} \) crossed by the network’s dynamics are stored in \( \text{output}^\mu \)

   proposal of an attractor code

2. \( \forall \) data \( D_{\mu}^b \) to learn

   (a) if \( \text{output}^\mu \) is a cycle of period greater than \( \max_{cs} \) \( \Rightarrow \) compression process (cf. below): \( \text{output}^\mu \mapsto \text{cycle}^\mu \);  
   (b) if \( \text{output}^\mu \) is a cycle of period lesser than \( \min_{cs} \) \( \Rightarrow \) extension process (cf. below): \( \text{output}^\mu \mapsto \text{cycle}^\mu \);  
   (c) if a pattern contained in \( \text{cycle}^\mu \) is too correlated with any other patterns, this pattern is slightly modified to make it “original”;

3. the data set \( D_{\text{temp}} \) is created where \( D_{\text{temp}}^\mu = (\chi^\mu, \text{cycle}^\mu) \)

learning the information

4. using an iterative supervised learning algorithm, the data set \( D_{\text{temp}} \) is tentatively learned a limited number of time steps;

5. if \( \forall \) data \( D_{\mu}^b \), the network iterates through valid limit cycle attractors \( \Rightarrow \) finished  
else goto 1

Compression process

1. \( \text{output}^\mu = \{\varsigma^{\mu,1}, \ldots, \varsigma^{\mu,\max_{cs}}, \ldots\} \) is a cycle of period greater than \( \max_{cs} \) (it could even be chaotic);  

2. \( \text{cycle}^\mu \) is generated by truncating \( \text{output}^\mu \): \( \text{cycle}^\mu = \{\varsigma^{\mu,1}, \ldots, \varsigma^{\mu,p_e}\} \)  
   (the compression period \( p_e \in [\min_{cs}, \max_{cs}] \) is another parameter).

Extension process

1. \( \text{output}^\mu = \{\varsigma^{\mu,1}, \ldots, \varsigma^{\mu,q}\} \) with \( q < \min_{cs} \);  

2. \( \text{cycle}^\mu \) is generated by duplicating \( \text{output}^\mu \): \( \text{cycle}^\mu = \{\varsigma^{\mu,1}, \ldots, \varsigma^{\mu,q}, \varsigma^{\mu,1}, \ldots\} \)  
   such that \( \text{size(\text{cycle}^\mu)} = p_e \), where \( p_e \in [\min_{cs}, \max_{cs}] \) is the extension period, another parameter, randomly given or fixed;

3. to make the cycle original, all the duplicated patterns are slightly modified
However, it has to be noted that this complete learning mechanism implicitly supplies the network with an important robustness to noise. First of all, the coding attractors are the ones naturally proposed by the network. Secondly, they need to have large and stable basins of attraction in order to resist the addition of new information for this reason, the parameters $ln$ and $ln_b$ have been set to 0 for all the experiments.

3 Experiments and Results

Experiments have been performed using the methodology as developed in previous chapters. First, the performance of this algorithm is quantified both in terms of the storing capacity and the noise robustness of the learned cycles. Then, the weight distribution is observed. Finally, dynamical analyses are performed on learned networks to gain a better understanding of the learning impact. These dynamical analyses are performed first by computing the mean Lyapunov exponent of the learned network$^2$, then by observing power spectra and return maps.

These results have been compared with the results previously obtained from the supervised version of this iterative Hebbian learning algorithm.

3.1 Performance of this algorithm

Figure 8.1 compares the number of information that can be learned in RNNs’ limit cycle attractors through the supervised and the unsupervised algorithms. The number of iterations required to learn the specified data set is plotted in function of the size of this data set. Each iteration represents one hundred weight modifications defined in Equation 6.13. The data set is composed of stimuli associated with cycles of size-2.

As expected, the unsupervised learning outperforms its supervised counterpart: the storing capacity is enhanced by a factor larger than six.

Furthermore, the performance of the unsupervised algorithm could still be enhanced since the limit cycles attractors have been explicitly restrained to size-2 to allow easy comparisons with the supervised algorithm. However, in the unsupervised case, the cycles’ size can be of any length within a range $[\min_{cs}, \max_{cs}]$, specified by the user before learning.

The six plots appearing in Figure 8.2 compare the robustness to noise obtained in networks learned through the supervised and the unsupervised algorithms. They show the Hamming distance between the stored sequence and the recovered sequence in function of the noise injected in the external stimulus $(m_{\text{ex}})$ and in the initial states $(m_{\text{in}})$. Again, the unsupervised learning algorithm considerably improves robustness since the network “decides on its own” on its limit cycles attractors. For instance, when learning 25 period-4 cycles, in the unsupervised case (Figure 8.2(d)), stored sequences

$^2$The mean Lyapunov exponent is obtained by computing a huge number of times the biggest Lyapunov exponent for this network (when it is presented with a random data) and by averaging these values. The computation of the Lyapunov exponent follows the method originally proposed by Wolf (Wolf et al., 1984).
Figure 8.1: This figure compares the encoding capacity of period 2 cycles between the unsupervised (blue curve with filled circles) and the supervised (red curves with empty circles) Hebbian learning algorithms. This also shows the number of iterations required. $N=25, \varepsilon_s = \varepsilon_b = 0, w_{ij} = 10$

are very robust to noise, while in the supervised case (Figure 8.2(c)), content addressability is no longer observed: by adding a tiny amount of noise, the correlation between the stored sequence and the recovered one goes to zero (the Hamming distance is equal to 50).

These figures also show that in the unsupervised case the external stimuli play a stronger role in indexing the stored data.

### 3.2 Weight distribution after learning

Figure 8.3 shows the weight distributions of learned networks when the learning task consists of storing an increasing number of stimuli in size-2 limit cycle attractors.

The first observation is that as the learning task is strengthened, the weight distributions are widened and the auto-connections become more and more negative. Similar trends have been seen with the supervised Hebbian learning algorithm.

However, the learning impact on the stimulus’ weight distribution shows a strong difference for the two algorithms. In both cases, their initial values have arbitrarily been set to 10, which has proven to give good results in terms of efficiency and convergence time. In the supervised case (Figure 7.4), the weight stimuli were significantly decreased during the learning (from their initial value set to 10, to values slightly above zero). Here, this decreasing does not appear: weight stimuli remain huge, even after extensive learning. This clarifies the results from the previous subsection: when a learned stimulus is presented to the network, its impact is huge (according to its weights). Hence, there is strong chance to retrieve the associated limit cycle attractor.

### 3.3 Lyapunov exponent analysis and probability of chaos

While the precedent section has given results of the performance enhancement when using unsupervised learning, this section aims at studying the learning process’ impact on the network’s inner dynamics.
Figure 8.2: These plots show the obtained content addressability for the learned networks: the Hamming distance between the expected and the obtained sequence of patterns is plotted in function of both the initial overlap of the initial states pattern $m_0s$ and of the external stimulus $m_0b$. The results for the supervised Hebbian algorithm (right figures) are compared with the ones obtained for the unsupervised Hebbian algorithm (left figures). $N=100$. 
3 Experiments and Results

Figure 8.3: Examples of weight distributions after unsupervised Hebbian learning of an increasing number of data in size-2 limit cycles attractors. \( N = 100 \)
Figure 8.4 shows a quantitative analysis of the “chaoticity” of the learned networks: the mean Lyapunov exponent and the probability to have chaotic dynamics are computed in networks learned with different data set of period-2 cycles.

Again, huge differences appear between this algorithm and its supervised counterpart. Figure 7.6 showed how networks learned through the supervised algorithm were becoming more and more chaotic while the learning task was strengthened. In the end, the probability of falling in a chaotic dynamics was equal to one. Here, even after learning a huge amount of data, networks do not become fully chaotic. This is explained by the fact that this unsupervised algorithm is based on a process of trials, errors and adaptations which 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. The network is more and more constrained, and complex, but not fully chaotic.

Figure 8.4: These figures show the impact of the learning algorithm on the complexity of the global dynamics by plotting the mean Lyapunov exponent and the probability to have chaotic dynamics of learned networks for different learning sets. N=25.

### 3.4 Power spectrum and return map analysis

A more qualitative description of the chaotic regime encountered in these networks can be obtained from return maps and power spectra analysis which have already proved to be very informative and characteristic.

Chapter 7 has shown that three types of chaotic regimes recurrently appear in networks learned through the supervised Hebbian algorithm. By increasing the size of the data sets, those networks go from highly informative frustrated chaos (indicated by chaotic itinerancy among a small number of nearby competing limit cycle attractors) to uninformative deep chaos similar to white noise. In other words, having too many competing limit cycle attractors leads to unstructured dynamics (see Figure 7.9).

For the unsupervised learning case, the obtained chaotic dynamics appear to be very informative all the time (deep chaos is hardly obtained). Figures 8.5 show an example of the network’s inner dynamics while the external stimuli are slowly modified from one learned stimuli to another one in five iterations (inner dynamics are
revealed through the return maps and the power spectra). A frustrated intermittency chaos appears in-between the two limit cycles attractors (represented at both sides). The evolution of this frustrated chaos reveals a shift of the influence of the two nearby competing limit cycle attractors.

4 Conclusion

This chapter introduces an algorithm which holds a position between the random networks and the networks learned with the supervised learning algorithm. In the random networks, the data set is completely unspecified a priori. In contrast, the data set of the supervised algorithm is fully specified a priori. Here, only the external stimuli are specified a priori. The learning task consists of associating each external stimulus with a limit cycle attractor of the network's dynamics. Making the learning unsupervised leaves the semantics of the information unprescribed until the learning occurs. 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 identifiable. Such a learning task could be related to neurophysiological observations where it has been shown that the same learned external stimuli can lead to different neural dynamics after different learning phase (Skarda and Freeman, 1987).

Without any special tuning of the parameters, the unsupervised algorithm, compared to its supervised counterpart, leads to great enhancements both in the storing capacity and in the computational cost. Moreover, since the unsupervised algorithm is based on a mechanism of trials-errors-adaptations, larger basins of attractions are obtained for the learned data.

Dynamical observations have shown that networks learned with the unsupervised algorithm show less chaotic dynamics than networks learned with its supervised counterpart which has proven to bring a road to chaos. This stability is a consequence of the large basins of attractions brought by the mechanism of trials-errors-adaptations. Moreover, when chaotic dynamics are encountered, chaos usually appears structured and meaningful, revealing the presence of nearby competing limit cycle attractors.
Figure 8.5: These figures show the 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)) while the external stimulus is slowly modified from one learned stimuli to another one. The 25 neurons network has learned 15 stimuli in period 4 cycles with the unsupervised learning algorithm.
Chapter 9

The frustrated Chaos

1 Introduction

The goal of this thesis is to try to find a way to store information in dynamical attractors of recurrent neural networks (RNNs). For that purpose, different learning algorithms have been proposed and tested. The methodology followed has led us not only to test the encoding capacities, but also to perform quantitative and qualitative dynamical analyses of the learned networks.

Learned networks have been shown to be fully stable in two cases. Firstly, after using the Backpropagation Through Time (BPTT) gradient based learning algorithm. Secondly after Hebbian learning of static patterns. This has been explained in the following way: since the BPTT algorithm works globally, it has a lot of difficulties to overcome local bifurcations. Hence, only stable networks can be learned. For the Hebbian learning of static patterns, this algorithm works locally in phase space and leads to the increased stability of learned patterns. Therefore this algorithm leads to stable networks.

In other cases, different types of network have shown chaotic dynamics. This was the case in random RNNs, in RNNs learned through the supervised Hebbian algorithm (in order to associate cyclic patterns) and in RNNs learned through the unsupervised Hebbian algorithm.

About a decade ago, chaotic itinerancy was proposed as a universal dynamical concept in high-dimensional systems (Ikeda et al., 1989), (Kaneko, 1992), (Tsuda, 1992). In (Kaneko and Tsuda, 2003):

“Chaotic itinerancy can be described by an itinerant motion among varieties of ordered states through high-dimensional chaotic motion.”

This type of informative and structured chaos has often been observed in our networks. More recently, Bersini has demonstrated that for some specific weight configurations, it was possible to drive small RNNs in chaotic dynamics (Bersini and Calenbuhr, 1997), (Bersini, 1998) and (Bersini and Sener, 2002). This chaos has been identified as originating from a physical frustration phenomenon and therefore 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.”

This chaos appears to be very related to the phenomenon of chaotic itinerancy. In this work we have chosen to refer to the appearance of this type of chaos as frustrated chaos. The aim of this chapter is to clarify this choice and to give more evidence of the presence of frustrated chaos in RNNs. The next section reviews the different chaotic dynamics encountered all along the preview chapters. A more detailed explanation of the frustrated chaos is given in the third section.

2 Summary of the chaotic dynamics encountered

All along the previous chapters, chaotic dynamics has been widely observed in RNNs. Different kinds of chaotic dynamics have been identified, using different dynamical tools (principally, by computing the biggest Lyapunov exponent and by plotting return maps, power spectra and output signals). This section highlights the fact that the kind of chaos obtained in RNNs is highly related to the parameters of this RNN.

2.1 Chaos in random RNNs

In Chapter 4 and 7 we have seen that in random networks, the network’s size is a very critical parameter for the kind of chaos encountered. In large random RNNs, on the one hand, networks usually are fully chaotic, with a chaos very similar to white noise: their attractors bear no structural information. On the other hand, small RNNs are barely fully chaotic and different kinds of chaotic dynamics have been observed: chaos obtained through a quasi-periodic road, chaos obtained through a period doubling road, and point-intermittent chaos. Moreover, it has been observed that this last kind of chaos is made of a brief itinerancy of nearby limit cycle attractors. Hence it has been identified as a frustrated chaos.

In (Bersini and Sener, 2002), frustrated chaos was obtained for a small RNN having a specific weight configuration. Here, it has been demonstrated that the frustrated chaos is in fact a very generic regime for small RNNs having a high encoding capacity. In these networks, in-between the numerous limit-cycle attractors, the dynamics hesitates between the nearby limit cycles in an almost unpredictable way, showing the phenomenon of frustration.
2.2 Chaos in Hebbian learned RNNs

2.2.1 RNNs learned through the supervised Hebbian algorithm

In Chapter 7, we have shown that supervised Hebbian learning of cyclic patterns leads the network to become more and more chaotic as the data set increases. Furthermore, we have observed that the kind of chaos encountered in the learned networks was highly related to the learning task. This led us to propose a classification of the different kinds of chaotic dynamics, in function of the learning tasks. This is shown in Table 7.1.

For a large data set, the network becomes highly saturated, and more and more chaotic dynamics appears. The chaos seems to be almost similar to white noise, i.e. bearing no structure. In the end the network is no longer able to manage anything and turns out to be fully and strongly chaotic. For a small data set, before reaching critical capacity, frustrated chaotic dynamics is usually observed in between these attractors.

2.2.2 RNNs learned through the unsupervised Hebbian algorithm

Frustrated chaotic dynamics has been observed both in random RNNs and in networks learned through the supervised Hebbian algorithm. However, in both cases, the network’s chaotic dynamics was composed of other types of chaos.

In Chapter 8, an unsupervised Hebbian learning algorithm has been proposed to learn the network to associate external stimuli with limit cycle attractors which are not specified a priori. This more plausible learning task has given very promising results in terms of encoding capacity. From a dynamical point of view, one characteristic of this algorithm is that it does not lead the network to become fully chaotic, even for large data sets. Moreover, this chaotic dynamics almost all the time seems to be originating from a frustration phenomenon.

3 Description of the frustrated chaos

3.1 Qualitative analyses

In previous chapters we have seen that frustrated chaos is characterized by very informative and structured return maps and by power spectra revealing the presence of nearby limit cycle attractors. Examples of these analyses can be seen in Figures 4.12, 7.9(c) and 8.5. These figures have been obtained by using ambiguous external stimuli, i.e. external stimuli standing in between learned external stimuli. By slightly modifying the external stimulus, the dynamics is trapped in a nearby limit cycles attractor.
3.2 Quantitative analyses

3.2.1 Probabilities of nearby limit cycles

Frustrated chaos is characterized by itinerancy among brief appearances of nearby limit cycles. To gain a better understanding of this chaos, this section aims at quantifying the presence of the nearby limit cycles in the chaotic motion.

Figure 9.1 compares a deep 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 5 data in limit cycles attractors of size 10 by using the supervised Hebbian algorithm. This algorithm hardly constrains the network and, as a consequence, chaotic dynamics appears very uninformative and unstructured: by shifting the external stimulus from one attractor to another one, the chaos in-between has lost any information concerning these two limit cycle attractors. In contrast, when coding the same number of information using the unsupervised algorithm (figure on the right), when driving the dynamics by shifting the external stimulus from one attractor to another one, the chaos encountered on the road appears much more structured: the strong presence of the nearby limit cycles can be clearly observed, shifting progressively from one attractor to the other one.

3.2.2 Spectrum of Lyapunov exponents

A dynamical system is characterized by a spectrum of Lyapunov exponents, their number being equal to the size of the dynamical system. The dynamics of the system is roughly characterized by its biggest Lyapunov exponent (see Section 5.4.3). As an example, chaotic dynamics is characterized by the presence of a positive Lyapunov exponent.

However, the other Lyapunov exponents can give more detailed information about the structure of the dynamics. As an example, strong chaos or hyper-chaos is characterized by the presence of more than one positive Lyapunov exponent (Rössler, 1983). Frustrated chaos is made of chaotic itinerancy, described by the balance between the attraction to low-dimensional motion (when the trajectory is nearly in one limit cycle attractor) and its escape from it. It is highly probable that this structured chaos has a characteristic Lyapunov spectrum, made of some negative exponents (attracting directions in the trajectories), many null Lyapunov exponents (the trajectory is neither attracted nor repulsed, instead it wanders in a low-dimensional motion) and one (or more) positive Lyapunov exponent (responsible for the chaotic motion and for the escape of the low-dimensional motion). However, this still needs to be analyzed and is the subject of future works.

3.3 Frustrated chaos related to chaotic itinerancy

The main differences between the frustrated chaos and the very similar and well-known chaotic itinerancy (Kaneko and Tsuda, 2003) lies in the way of obtaining it as
3 Description of the frustrated chaos

Figure 9.1: Probability of presence, for one neuron, of the nearby limit cycle attractors in a chaotic dynamics (x-axis). By slowly shifting the external stimulus from an external stimulus previously learned (region (a)) to another stimulus learned (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. N=100.
well as in the possible characterization of the dynamics in terms of the encoded attractors. If all those very structured chaotic regimes are characterized by strong cyclic components among which the dynamics randomly itinerates, it is in the transparency and the exploitation of those cycles that lie the key differences. As far as we know, the low-dimensional motions which compose the chaotic itinerancy have no predefined meaning: they are merely observed as being very characteristic. In contrast, in the frustrated chaos, chaotic dynamics exhibit itinerancy among low dimensional motions which are related to the limit cycles attractors learned in the RNN (as shown in Figure 9.1). Those cycles are the basic stages of the road to chaos. It is by forcing those cycles in the network and by tuning the connection parameters that the chaos finally appears. Hence, the frustrated chaos is expected after learning. Such a road to frustrated chaos has been obtained through unsupervised Hebbian learning.

4 Conclusion

All along this thesis, chaos has often been observed as a by-the-way regime appearing in learned networks or in networks having a high encoding potential. The more information the network has to store in its attractors the more chaotic it spontaneously tends to behave. Chaos is in fact the biggest pool of potential cyclic attractors.

One chaotic regime encountered has been identified as the frustrated chaos, originally discovered in (Bersini and Calenbuhr, 1997). This chaos is characterized by unpredictable itinerancy among nearby learned limit cycle attractors. To demonstrate the presence of the “frustrated chaos”, the presence of the nearby learned limit cycle attractors in chaotic dynamics is computed in this chapter. This frustrated chaos is obtained by 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.
Chapter 10
Conclusion

This thesis proposes the use of recurrent connectionist networks by associating them to a new type of learning tasks, i.e. learning information in their dynamics. Every information is composed of the external stimulus and the internal signal generated by the network’s dynamics. The connectionist interest in this model lies in the interaction between the external stimulus and the initial state of the network to produce the expected signal on its internal dynamics.

This learning task has two roots. Its first root lies in the work performed by Hopfield more than twenty years ago. Hopfield has shown that by following an Hebbian prescription rule, the network is able to learn information in fixed point attractors of the network’s dynamics (Hopfield, 1982). However, very poor results were obtained (Amit et al., 1985). The use of external stimuli makes our model stand apart. Moreover, in our model, not only fixed point attractors but also limit cycle attractors can provide meaningful information. These two modifications have proven to boost the network’s encoding capacity compared to the classical Hopfield model.

The second root of the learning task we developed lies in the seminal observations realized more than fifteen years ago by Skarda and Freeman. Together, they 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 (Skarda and Freeman, 1987). These observations have led us to use limit cycle attractors as a valuable source of information.

However, no straightforward direction was found to store information in recurrent neural networks through complex dynamics. Hence, we decided to replace this missing direction by a straightforward methodology, namely the focus of our analysis not being only the direct results of the networks’ encoding capacities, but also the dynamical observations of these networks before and after each proposed learning procedure. This methodology was to help us in finding engineering and/or cognitivist applications. However, it appeared that currently no software exists generic enough to provide different kinds of dynamical tools for any kind of neural architecture. Being confident in the future of our research, we decided to build our own tool, and to make it as generic as possible to enable future work. The created platform has been
named the “Neural Development Kit” after the second release. All development time and investments are beginning to pay back.

The first part of our work consisted of analyzing the different dynamics spontaneously occurring in RNNs. For this first analysis, the weight connections linking neurons together were randomly chosen from different kinds of weight distributions and network parameters. Because these neurons’ activation was chosen to be continuous, any slight modification of an external stimulus led to a different output. Since the aim of the work is based on the possibility to robustly distinguish the outputs obtained from different external stimuli, a symbolic quantization on the outputs was introduced. This symbolic quantization was obtained by adding a filter layer.

It has been shown that these networks, when randomly chosen, have a great number of dynamical attractors, passing from one to another by modifying external stimuli and depending on their initial states. But more importantly, the importance of chaos in this encoding process has been stressed. Different types of chaos have been identified: chaos obtained by period doubling, chaos obtained by quasi-periodicity, and another type of chaos revealing the strong presence of nearby attractors and therefore appearing to be due to a mechanism of frustration. Results clearly showed that the wider the encoding potential for a RNN, the more chaos prevails as a natural and spontaneous dynamical regime. Chaos boosts the encoding capacity. Referring to Skarda and Freeman, this makes sense of the world. Referring to Nietzsche, this gives birth to dancing stars. These promising analyses stimulated us to continue in this direction.

So far, storing capacity has been investigated in the absence of any practical way of storing information known a priori. Without any learning algorithm, this encoding potential remains useless, compromising the use of these nets for any practical application. In order to use dynamical limit cycles attractors to encode information, learning algorithms are necessary. The first approach followed was to use standard supervised learning algorithms: when fed by an external stimulus and when its initial state is not too far from some configuration, the network’s dynamics is expected to live in a specific limit cycle attractor.

The first supervised algorithm investigated was the gradient based backpropagation through time learning algorithm (BPTT). This gave poor results and time consuming computations due to the fact that the gradient descent based approach is often stopped by the numerous bifurcations encountered during the descent. As a consequence, learned networks appear to be entirely stable with 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.

The second supervised algorithm tested followed a quasi-unanimous view shared by neural net researchers, i.e. 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 fulfilling the needs of our model: to encode cyclic information by relying not only on the internal state but also on the external stimulus. 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.
A modification to the learning and retrieving algorithm was proposed in this thesis, consisting of using external stimuli to enhance performance. 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 parallel with this enlargement, both a slight increase of the storing capacity and an improvement of the robustness to noise have been observed.

In order to enable dynamical observations of the learned networks, the supervised iterative Hebbian based learning algorithm was adapted to continuous state neurons. Expected results have been obtained when coding external stimuli in fixed point attractors of the network’s dynamics: no chaotic dynamics showed up for ambiguous stimuli (i.e. for stimuli standing in between learned stimuli). However, an impressive observation has been obtained when using this algorithm to encode information in limit cycle attractors of the network’s dynamics. The more information stored, the more chaos becomes unavoidable on the road 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 input leads to ambiguous dynamics (Kelso et al., 1995).

Even though the results obtained using the iterative supervised Hebbian learning algorithm were not so bad, they remained far below the potential expected from these powerful connectionist networks. Supervised learning seemed to be too constrained. Hence, in a second approach, an unsupervised learning task, biologically more plausible, was introduced: the network has to learn to react to an external stimulus by cycling through a sequence which is not specified a priori. Making the learning unsupervised leaves the semantics of the information unprescribed until the learning occurs. 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 net terms by several authors (among others Varela et al., 1991, Erdi, 1996, Tsuda, 2001). One operational form has achieved great popularity as a neural net implementation of statistical clustering algorithms (Kohonen, 1982), (Grossberg, 1992). Moreover, this perspective stays in line with the dynamical hypothesis. The coding scheme is no longer located at the neuron level (each neuron having its predefined meaning), instead the interpretation is casted at the dynamical level.

An unsupervised iterative Hebbian learning algorithm was developed to learn the network. Compared to its supervised counterpart, huge enhancements both in the storing capacity and in the computational cost have been observed. Moreover, since the unsupervised algorithm is based on a mechanism of trials-errors-adaptations, larger basins of attractions are obtained for the learned data. As a consequence, learned data
show good tolerance to noise.

From a dynamical point of view, considerable differences was observed between the two types of Hebbian algorithms. The iterative supervised Hebbian learning can be seen as an alternative road to chaos, the background chaos spreads widely and adopts a very unstructured shape similar to white noise. In contrast, the unsupervised learning, by being more “respectful” of the network intrinsic dynamics, maintains much more structure 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”.

The last chapter was dedicated to some extra analyses of the chaotic dynamics encountered, and more particularly of the frustrated chaos. As stated in (Bersini, 1998), the frustrated chaos is characterized by unpredictable itinerancy among nearby learned limit cycle attractors. Therefore, to demonstrate the presence of this kind of chaos in our networks, the probability of presence in chaotic dynamics of the nearby learned limit cycle attractors was computed. The plots were obtained by 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. Using these results, the predominance of this highly informative chaos in networks learned through the unsupervised Hebbian algorithm was demonstrated. This kind of structure and “informative” chaos has made several authors conclude that this kind of chaos 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 (Skarda and Freeman, 1990a), (Rodriguez et al., 1999).

The work presented in this thesis suggests that RNNs could be used in practical applications to robustly encode information in the limit cycle attractors of the network’s dynamics. By relaxing the constraint imposed by Hopfield to store information in fixed point attractors, and by using external stimuli to index the stored information, the first enhancements were obtained, i.e. the set of memory bags have been considerably enhanced. A second wave of enhancements was obtained by relaxing the constraints imposed on the learning task and by relying on an unsupervised algorithm which makes more sense when adopting both a biological and a cognitive perspective. An extra “gift” of this learning task is the incorporation of informative and highly structured chaotic dynamics in the spontaneous regime obtained for ambiguous stimuli. This highly informative chaos may provide the basis for flexibility, adaptiveness, and the trial–and–error coping that enables the nervous system’s interaction with an unpredictable and ever-changing environment.

Among the numerous possible future works, only the most pertinent and relevant have been stated here.

In order to get a better understanding of the chaotic regimes encountered, two approaches can be followed. First, we believe that interesting results are obtained from the analyses of the Lyapunov spectrum, especially for the frustrated chaos where numerous null Lyapunov exponents are expected. Second, all results presented here
are only based on temporal analyses - spatial investigations could give interesting results. In line with neurophysiological observations, spatial coherence is expected for the frustrated chaos.

From an engineering point of view, the unsupervised learning mechanism proposed should be tested and compared to the classical clustering techniques.

After unsupervised learning, the learned network gives its own “interpretation” to the external stimuli and a feedforward network could be added to the output to process the information in order to make it more meaningful. A feedforward network in the input is expected to enhance the performance. These modifications will lead to a hybrid architecture constituted of three networks, or three “neuro-modules”. The platform developed is ready for such a complex architecture.

In all the learning tasks presented in this thesis, memories were viewed as static features. After learning, the information is coded in an immutable way which does not seem to be biologically plausible. A slight modification on the unsupervised algorithm should be performed to enable online learning and to provide dynamical memories. In this perspective, memories are no longer stored in immutable fixed points or limit cycle attractors perfectly identifiable and therefore considered as recoverable features. Instead, memories are re-created, always with differences, at the same time that they are retrieved. Hence memory becomes an active process.
Appendix A

Development of a Neural Development Kit

1 Introduction

The numerous models of artificial recurrent neural networks and the lack of convincing applications for them provide a clear picture of the state of the art in this field. These recurrent networks seem so powerful that they appear difficult to control and to learn. The author of this thesis shares the widely accepted idea that a better understanding of non-linear dynamical phenomena occurring in fully recurrent neural networks should be able to help to find engineering and/or cognitivist applications. Therefore the dynamical impact of different learning algorithms on fully connected recurrent neural networks have been studied in this thesis.

After reviewing the existing neural softwares, it appeared that there is no platform generic enough to provide different kinds of dynamical tools. Therefore, we decided to build our own tool, the “Neural Development Kit”. To enable and prepare future work, we decided that this platform should also enable the development and the testing of any kind of neural architecture.

The main goal for this development kit was to allow customization at two levels. The first level consisted of runtime customization, meaning the implementation of a tool which is highly generic and which allows modifications of the common parameters. Moreover, to ease its manipulation, we decided to focus on an enhanced graphical user interface. This interface was to enable the creation and management of any network architecture, to plot different dynamical results, to configure all (or most) of the network’s parameters in an easy an intuitive way, etc. This graphical part gives a meaning to the “X” in the XNDK. The second part of the customization is on the code design. The code is thought to be generic and extension aware as much as possible. This leads to the anticipated and unanticipated cooperation and the interaction between different parts of the development kit.

Like for any other software, the development of a neural development kit is an incremental process. From the beginning we decided to follow the extreme program-
ming (XP) philosophy as much as possible (Beck and Andres, 1999). To a large extent, this philosophy is based on the interaction between different actors (mainly the client, the architect and the programmers), whereas the first release of the XNDK platform has been created only by the author and myself. However, with the creation of the second release, Utku Salihoglu joined the team, giving the XP philosophy more meaning.

2 Requirements

The common features and functionalities to be expected from a generic neural development kit can be found at two levels. First, there are numerous requirements from a user point of view. Second, requirements can be found at the code level, allowing the easy addition of new functionalities.

2.1 Runtime requirements

The main functionalities we wanted to include in the neural toolbox were the following:

- **design the network** the user must be able to create, modify and destroy neurons with a graphical user interface. This approach should also allow the combination of different networks, creating hybrid architectures;

- **apply learning policy** the user must be able to learn the network (or part of the network) using a well defined learning algorithm. Moreover, the parameters of this algorithm should be easy to configure;

- **manipulate parameters** the user must be able to modify all the parameters of the built network intuitively (weight connections, neuronal activators, update rule, etc.);

- **easy simulations** the user should be able to easily simulate the network and to visualize the network’s behavior;

- **dynamical analyses** the user should be able to perform different dynamical analyses on the built network and on its components;

- **statistical results** the user should be able to obtain statistics allowing to highlight global behaviors.

The use case diagram depicted in Figure A.1 summarizes the user interaction with the application.

2.2 Implementation requirements

Since research is an ongoing process, it is expected that this platform will never be completed. New functionalities will always be required. Principally, this leads to the two following requirements: the code must be available (not private) and must be as transparent as possible to allow easy modifications.
There are lots of tools on the market designed to manipulate neural networks. However, as far as we know there is no neural toolbox fulfilling the requirements mentioned in the last section. First of all, not one of the existing platforms enables dynamical analyses on different types of neural architectures. Also, no platform allows the easy incorporation of new types of learning algorithms. Finally, the code of the investigated neural toolbox is not generic enough to enable the type of manipulations required by our work. The sum of these observations justified the implementation of a new development kit.

3.1 Analysis of the runtime requirements

In this section, the numerous runtime requirements outlined in the last section are analyzed from a developers point of view.

**design the network** This has resulted in our focus on the development of an intuitive graphical user interface enabling the creation, the destruction and the manipulation of the network components. In order to enable the creation of a hybrid architecture, we have decided to add a special component in our network, i.e; the “subnet”, being a container composed of other subnets and neurons;

**apply learning policy** a library of learning policy must be created. The addition of a
new learning policy to the network has to be very simple and straightforward. The learning policy receives a reference to the network (the subnet) to learn through a simple manipulation of the graphical user interface;

**manipulate parameters** each network component must implement a function allowing access to all required configuration options;

**easy simulations** network or subnet simulations must be possible from numerous access points (from plot diagrams, statistic diagrams, etc.);

**dynamical analyses** a library must be implemented containing most of the dynamical tools (see below);

**statistical results** a library must be provided allowing the required statistics.

### 3.2 Analysis of the implementation requirements

In order to fulfill the implementation requirements, we decided to follow the object-oriented paradigm. All key concepts (activator, learning policy, etc.) have been defined using abstract classes, allowing any developer to add his own custom type transparently. The new type added is automatically taken into account thanks to the polymorphism mechanism.

Another requirement has been to allow any manipulation at a low level as well as at a high level (by "manipulation of concepts"). For this purpose, low level manipulations should be possible via a main entity, knowing every parameter of every object, organized on vectors for fast processing. High level manipulations should be possible relying on the object-oriented message paradigm.

### 4 Design

The development of an application using the object-oriented paradigm requires an extensive focus on its design. Different well-known design patterns have been implemented in our code in order to render it as generic as possible (a design pattern can represents a high-quality solution to a recurring problem in design). For further information on these designs, we strongly suggest to consult the famous book written by the “gang of four” (Gamma et al., 1995).

The application is divided in two major parts: the neural development kit (ndk) and the neural development interface (ndi). The neural development kit is the core of the application, handling the network management, the computations and the learning algorithms.

The main components of the core (ndk) are represented in Figure A.2, i.e. the brain, the subnet and the neuron. The brain is the central element of the core, containing the weight matrix, the external stimulus’ weights, the external stimulus’ values, the internal state’s values, as well as the connection matrix (determining if two neurons are connected together). The brain allows all the low level manipulations we expect.
Figure A.2: Main elements of the ndk: UML class diagram

However, a high level design has been added through the introduction of the subnet and the neuron. The brain uses the Singleton pattern, making it unique and giving it a unique access point.

The subnet is constituted of neurons and allows the distinction between different groups of neurons. It is also possible for a subnet to contain other subnets, allowing the design and the manipulation of any hybrid network architecture. The subnet architecture can be seen as a Composite pattern since it is composed of subnets and neurons (see Figure A.3).

The neuron class enables having direct access to all the properties of the neuron (its weight connections, its activator, etc.). This association between the neuron and its properties is well known as the state pattern. Moreover, the neuron has a reference to its subnet, enabling access to the father node in the composition graph and being primordial to apply the design pattern stated as the chain of responsibilities in (Gamma et al., 1995). The neuron possesses very local information, can handle some specific operations on it and can ask the subnet to handle the rest.

In contrast, subnets are not contained in the brain (except for the main subnet, i.e. the brain itself). Creating subnets through existing subnets allows avoiding the specification of the created subnet’s localization within the whole architecture. A simple simulation process of a subnet is described in the sequence diagram Figure A.4.

The ndi is plugged on top of the ndk, adds a graphical representation to it and implements the classical Model-View-Controller design pattern. Following this pattern, the Brain class is the Model, Neuron and SubNet classes are the controller and the ndiSubNet and the ndiNeuron classes are the View. Thanks to this pattern, views can be changed at runtime, without affecting the behavior of the core.

5 Practical aspects

The XNDK is currently in its third version. The whole implementation is done using C++/QT. C++ was adopted since its seemed to be the best compromise between performance and high level programming. QT is a high level library extending C++, providing multiple facilities and is principally being used for all the graphical user interfaces of the XNDK.
From the beginning, we decided to build a platform independent software (the first release was written in Java).

The current version of the platform contains more than one hundred classes and more than sixty thousands lines of code (without blank lines and comment lines).

Different dynamical analyses can be performed using the XNDK. These analyses are then visualized on diagrams, each diagram representing the dynamical results obtained either on individual neurons or on the global behavior of the subnet. These
diagrams are the following:

- the state versus time diagram, showing the output signal of the network (or the neuron);
- the power spectrum diagram;
- numerous types of bifurcation diagrams, enabling to see the road to chaos;
- the return map diagram;
- the iso map diagram;
- 3D realtime state diagrams

The different learning algorithms provided in the XNDK are the following:

- the classical backpropagation algorithm;
- the classical backpropagation through time algorithm;
- the classical Hebbian rule;
- the classical iterative supervised Hebbian algorithm;
- an adaptation of the iterative supervised Hebbian algorithm which takes into account the presence of external stimuli;
- an iterative unsupervised Hebbian algorithm

* contribution of the author.

Four different views of the XNDK are given in Figures A.5, A.7, A.7, A.8.

---

1 this is a 3D plot in which each axis stands for one neuron value. A point wanders in the phase space in realtime representing the variation of the state value of three neurons.
Figure A.5: View of the XNDK: a RNN of 5 neurons and a bifurcation diagram

Figure A.6: View of the XNDK: a RNN of 5 neurons and a realtime diagram
5 Practical aspects

Figure A.7: View of the XNDK: learning cycles using a supervised iterative Hebbian algorithm

Figure A.8: View of the XNDK: construction of a complex hybrid architecture composed of one feedforward network feeding an RNN, feeding in turn another feedforward network.
Bibliography


