UNIVERSITE LIBRE DE BRUXELLES
FACULTE DES SCIENCES
DEPARTEMENT D'INFORMATIQUE

Sélection Séquentielle en Environnement Aléatoire Appliquée à l'Apprentissage Supervisé

Thèse présentée par

Olivier Caelen

En vue de l'obtention du grade de

Docteur en Sciences

Septembre 2009

© 2009

Olivier Caelen

All Rights Reserved

Cette thèse a été écrite sous la supervision du Professeur Gianluca Bontempi. Les membres du jury sont :

- Prof. Luc Barvais (Université Libre de Bruxelles, Belgique)
- Prof. Mauro Birattari (Université Libre de Bruxelles, Belgique)
- Prof. Gianluca Bontempi (Université Libre de Bruxelles, Belgique)
- Prof. Bernard Fortz (Université Libre de Bruxelles, Belgique)
- Prof. Guy Latouche (Université Libre de Bruxelles, Belgique)
- Prof. Rémi Munos (Institut National de Recherche en Informatique et Automatique, Lille, France)

Résumé

Cette thèse se penche sur les problèmes de décisions devant être prises de manière séquentielle au sein d'un environnement aléatoire. Lors de chaque étape d'un tel problème décisionnel, une alternative doit être sélectionnée parmi un ensemble d'alternatives. Chaque alternative possède un gain moyen qui lui est propre et lorsque l'une d'elles est sélectionnée, celle-ci engendre un gain aléatoire. La sélection opérée peut suivre deux types d'objectifs.

- Dans un premier cas, les tests viseront à maximiser la somme des gains collectés. Un juste compromis doit alors être trouvé entre l'exploitation – se focalisant sur les alternatives qui semblent pour l'instant offrir un gain optimal – et l'exploration, tendant à maximiser les gains à plus long terme en testant les alternatives peu connues. Ce problème est couramment dénommé dans la littérature scientifique « multi-armed bandit problem » .
- Dans un second cas, un nombre de sélections maximal est imposé et l'objectif consistera à répartir ces sélections de façon à augmenter les chances de trouver l'alternative présentant le gain moyen le plus élevé. Les alternatives sont tout d'abord explorées de manière séquentielle et c'est ensuite l'alternative semblant, pour l'instant, optimale qui se voit désignée. Ce deuxième problème est couramment repris dans la littérature scientifique sous l'appellation « selecting the best » .

Dans ces problèmes de décision, la tâche consiste toujours à trouver le plus rapidement possible l'alternative optimale ayant le gain moyen le plus élevé. La sélection de type gloutonne joue un rôle important dans leur résolution et opère en choisissant l'alternative qui s'est jusqu'ici montrée optimale. Or, la nature généralement aléatoire de l'environnement rend alors incertains les résultats d'une telle sélection.

Dans cette thèse, nous introduisons une nouvelle quantité, appelée le « gain espéré d'une action gloutonne » , qui utilise la notion de « probabilité de faire une sélection correcte » (PCS) bien connue en simulation. Sur base de quelques propriétés de cette quantité, de nouveaux algorithmes permettant de résoudre les deux problèmes décisionnels précités seront proposés.

Ces problèmes de décision séquentielle se rencontrent dans de nombreuses situations de la vie courante. Une attention particulière sera ici

prêtée à l'application des techniques présentées au domaine de l'apprentissage artificiel supervisé. Un tel apprentissage consiste à construire – sur base d'exemples – une fonction prédictive d'un phénomène d'entrée/sortie inconnu. Une des étapes fondamentales de tout problème d'apprentissage supervisé est la sélection de modèles consistant à choisir, au sein d'une série de classes de modèles, celle à utiliser pour construire la meilleure fonction prédictive. Cette étape peut être réalisée via des méthodes de *cross-validation* qui testent chaque classe de modèles de manière séquentielle et où un seul test peut déjà demander un effort de calcul relativement grand. Afin d'accélérer la procédure de sélection de modèles, il faut donc trouver – en un minimum de tests possibles – la classe de modèles optimale. Si les différentes classes de modèles correspondent aux alternatives, il est alors possible d'utiliser les algorithmes exposés dans cette thèse.

La collaboration avec le service d'anesthésie de l'Hôpital Erasme nous a permis d'appliquer les algorithmes proposés dans cette thèse à des données réelles, provenant du milieu médical. Nous avons également développé un système d'aide à la décision dont un prototype a déjà été testé en conditions réelles sur un échantillon restreint de patients.

Abstract

This thesis focuses on the problems of sequential decision making in a random environment. At each step of such a decision-making problem, an alternative which generates a random gain must be selected from a set of alternatives each of which has a specific average gain. The selection process may follow two objectives :

- In the first case, the tests could aim to maximize the sum of gains collected. A fair trade-off must be struck between exploitation - focusing on alternatives that seem to offer a current optimal gain - and exploration, to maximize long term gain by testing unknown alternatives. This problem is commonly referred to in the scientific literature as "multi-armed bandit problem".
- In the second case, a maximum to the number of selections is imposed and the objective is to distribute these selections in order to increase the chances of finding the alternative with the highest average gain. The alternatives are first explored in a sequential manner and the most optimal alternative currently is designated. This problem is commonly called "selecting the best" alternative in the scientific literature.

In these decision problems, the task is to find the optimal alternative with the highest average gain as quickly as possible. The greedy selection plays an important role in the solution to these problems. It operates by selecting the alternative that will be proved by then to be the optimal since the start of the selection process. However, the generally random nature of the environment makes the results of such a greedy selection uncertain.

In this thesis, we introduce a new quantity called the "expected gain of a greedy action", which uses the notion of "Probability of Correct Selection" (PCS), which is well-known in simulation literature. Based on some properties of this quantity, new algorithms to solve the two decision problems described above will be proposed.

These sequential decision problems are found in many real-life situations. A particular attention will be paid to the application of the techniques presented here in supervised machine learning. Such a learning consists of building a predictive function of an unknown input/output phenomenon on the basis of examples. One of the fundamental steps in supervised learning is model selection which chooses, in a series of classes of models, the one to

be used to build the best predictive function. This step can be performed via computationally intensive methods of cross validation which test each class of models sequentially. To speed-up the process of model selection, the optimal class of models must be found with a minimum of possible tests. If the different classes of models correspond to the alternatives, it is then possible to use the algorithms described in this thesis.

Collaboration with the Department of Anaesthesiology at the Erasme Hospital, Brussels allowed us to apply the algorithms proposed in this thesis to real data in a medical environment. For this purpose, we have also developed a decision support tool, a prototype of which was tested in real conditions on a small sample of patients.

Remerciements

Cette thèse est le résultat d'un travail passionnant de plusieurs années qui m'a permis d'aborder l'étude d'un domaine de recherche extrêmement intéressant.

Tout d'abord, je tiens à remercier tout particulièrement le Professeur Gianluca Bontempi sans qui tout ceci n'aurait pas été possible. Je le remercie pour tous ses précieux conseils, ses encouragements, son aide à la rédaction des articles scientifiques et sa disponibilité. Je le remercie encore de m'avoir mis en contact avec des horizons de recherche très enrichissants et diversifiés tels que les milieux médicaux et financiers.

Je remercie également les membres du jury d'avoir accepté de commenter ce travail : le Professeur Luc Barvais, le Professeur Mauro Birattari, le Professeur Gianluca Bontempi, le Professeur Bernard Fortz, le Professeur Guy Latouche et le Professeur Rémi Munos.

Je souhaiterais ensuite remercier les membres du Machine Learning Group de l'ULB : Abhilash, Benjamin, Catharina, Kevin, Mathieu, Patrick, Yann-Aël avec qui les contacts ont été très chaleureux. Nos conversations très enrichissantes ont permis l'échange d'idées dans des domaines très variés.

Je remercie mes collègues de bureau pour les bons moments passés en leur compagnie durant toutes ces années : (par ordre chronologique) Rachel, le Professeur Machgeels, Liliana, Abhilash, Olivier.

Cette thèse a été réalisée en collaboration avec le service d'anesthésie de l'Hôpital Erasme. Je remercie le Professeur Luc Barvais ainsi que tous ses collaborateurs pour leurs précieux conseils. Je tiens également à remercier Monsieur François Clement avec qui la collaboration fut fort agréable et fructueuse.

Dans le cadre de cette thèse, j'ai passé six mois au *Laboratorio di Intelligenza Artificiale e Robotica del Politecnico* à Milan. Je remercie le Professeur Andrea Bonarini et son équipe pour leur chaleureux accueil et leur collaboration scientifique.

Plusieurs personnes ont lu ce manuscrit dans différents stades et ont donné leur avis. Je remercie le Professeur Devillers, Yann-Aël, Patrick, Olivier et Abhilash pour leurs aide et commentaires.

Je remercie ma famille pour leur soutien moral et leurs encouragements durant toutes ces années. Je remercie en particulier mes parents pour leur

aide et pour avoir finalement compris ce qu'est μ_g.

Pour terminer, je tiens à remercier tout particulièrement ma compagne Alexandra pour l'amour et le soutien qu'elle m'a donnés. Je sais que cette période de rédaction a aussi été difficile pour elle. Je la remercie de m'avoir supporté (dans les deux sens du terme).

Financement

Le travail présenté dans cette thèse a été financé par le projet First Europe Objectif 1 intitulé « Data mining prédictif en anesthésie intraveineuse informatisée en vue de certification » de la Région Wallonne et par le projet intitulé « OASIS » financé par la Politique Scientifique Fédérale Belge.

Table des matières

1 Introduction **1**
- 1.1 La sélection de l'optimum avec un budget de tests limité . . . 2
- 1.2 Le *multi-armed bandit problem* . . . 3
- 1.3 Formalisation du problème . . . 4
- 1.4 Sélection séquentielle et sélection de modèles . . . 5
  - 1.4.1 L'application des méthodes d'apprentissage artificiel en anesthésie . . . 7
- 1.5 Les principales contributions de cette thèse . . . 7
- 1.6 Structure de la thèse . . . 9
- 1.7 Publications . . . 10
- 1.8 Notation . . . 11

2 Méthodes de sélection de modèles en apprentissage supervisé **15**
- 2.1 Formulation du problème d'apprentissage supervisé . . . 15
  - 2.1.1 Le problème de régression . . . 19
  - 2.1.2 Exemple du problème d'identification paramétrique . . 19
- 2.2 Le compromis estimation-approximation . . . 21
  - 2.2.1 Illustration du compromis estimation-approximation . 23
- 2.3 L'identification structurelle . . . 25
- 2.4 Sélection de modèles par minimisation de l'erreur de généralisation 28
  - 2.4.1 L'erreur de généralisation . . . 28
  - 2.4.2 Le compromis biais-variance . . . 30
  - 2.4.3 L'estimation de l'erreur de généralisation . . . 31
- 2.5 La sélection de variables . . . 34
- 2.6 Les familles d'ensembles d'hypothèses . . . 36
  - 2.6.1 Les familles d'hypothèses de type global . . . 36
  - 2.6.2 Les familles d'hypothèses de type diviser pour conquérir 37
- 2.7 Conclusions . . . 39

3 Problèmes de sélection en environnement aléatoire **41**
- 3.1 Formalisation des problèmes de sélection séquentielle dans un environnement aléatoire . . . 42

3.2 Les stratégies d'échantillonnage pour la sélection de l'optimum 43
3.2.1 La sélection de sous-ensembles d'alternatives 43
3.2.2 Les procédures de sélection en deux étapes d'échantillonnage 46
3.2.3 Les procédures combinées pour l'optimisation de la sélection en deux étapes d'échantillonnage 49
3.2.4 Les procédures séquentielles pour l'optimisation de la sélection 51
3.3 Le *multi-armed bandit problem* 58
3.3.1 Formalisation du problème 58
3.3.2 Les stratégies semi-uniformes 60
3.3.3 Les stratégies basées sur les intervalles de confiance . . 64
3.3.4 Les stratégies probabilistes 65
3.3.5 La stratégie d'indice de Gittins 66
3.4 Conclusions 67

4 Le gain espéré d'une action gloutonne – contributions 69
4.1 Définition du gain espéré d'une action gloutonne 70
4.1.1 Le gain espéré d'une action gloutonne lorsque $K = 2$. 74
4.1.2 Le gain espéré d'une action gloutonne dans la sélection de modèles 74
4.2 Etude sur l'évolution de la quantité μ_g 77
4.2.1 L'évolution de μ_g lorsque $K = 2$ 77
4.2.2 L'évolution de μ_g lorsque $K > 2$ 84
4.3 Calcul de la probabilité de sélectionner l'alternative k par Monte Carlo 91
4.4 Estimation du gain espéré d'une action gloutonne 94
4.4.1 L'estimateur *naïf* 95
4.4.2 L'estimateur par *plug-in* 95
4.4.3 L'estimateur par *mise de côté* 96
4.4.4 L'estimateur par *leave-one-out* 97
4.4.5 Expériences d'estimation de μ_g 98
4.5 Conclusions 102

5 Les stratégies de sélection de l'optimum – contributions 103
5.1 La stratégie optiK2Estim ($K = 2$) 104
5.2 La stratégie *PCSsearch* ($K \geq 2$) 105
5.3 La stratégie exploreMaxμ_g ($K \geq 2$) 107
5.4 Expérimentations et discussion 110
5.4.1 Problèmes créés de manière synthétique 110
5.4.2 Problèmes de sélection de modèles locaux 115
5.4.3 Discussion 117
5.5 Conclusions 118

6 Algorithmes semi-uniformes pour le problème du *bandit* – contributions 121
6.1 Reformulation du problème lorsque l'information sur l'état du système est parfaite ou imparfaite 122
6.2 La stratégie ϵ-PCSgreedy 124
6.3 La stratégie DP-*greedy* 125
6.3.1 L'algorithme semi-uniforme DP-*greedy* dans le cas PSI 126
6.3.2 L'algorithme semi-uniforme DP-*greedy* dans le cas ISI 129
6.4 Expérimentations et discussion 131
6.4.1 Expériences sur la stratégie ϵ-PCSgreedy 131
6.4.2 Expériences sur la stratégie DP-*greedy* 133
6.5 Conclusions 135

7 Application des algorithmes d'apprentissage supervisé en anesthésie – contributions 137
7.1 L'anesthésie intraveineuse à objectif de concentration 139
7.2 Formalisation du problème 143
7.2.1 L'ensemble d'apprentissage 145
7.2.2 La sélection de variables 147
7.3 Le logiciel BisPrediction 149
7.4 Accélération de la procédure de sélection de variables avec exploreMaxμ_g 151
7.5 Conclusions 155

8 Conclusions 157
8.1 Perspectives futures 159

A Résultats expérimentaux de *PCSsearch* et exploreMaxμ_g réalisés sur des données synthétiques 163

B Les hypothèses linéaires 169
B.1 Le least-squares 169
B.2 La PRESS 171

C Définition analytique du regret cumulé 173

D La programmation dynamique 175

E L'inégalité de Hoeffding 177

F La loi de distribution khi-carré 179

G Le maximum d'un ensemble de variables aléatoires 181

Chapitre 1

Introduction

Ce travail étudie les problèmes de décisions séquentielles prises dans un environnement aléatoire. Nous considérons plus spécifiquement les deux problèmes suivants : le « multi-armed bandit problem » et le « problème de la sélection de l'optimum avec un budget de tests limité » . Une attention particulière sera également prêtée à l'application de ces méthodes en apprentissage artificiel afin de développer un système d'aide à la décision en anesthésie.

Tout un chacun se trouve quotidiennement confronté à des problèmes dans lesquels un choix doit être opéré parmi une série d'alternatives. Il faut alors décider quelle alternative sélectionner en vue de maximiser un objectif dépendant du type de problème rencontré.

Certains de ces choix auront relativement peu d'impact sur le long terme ; comme par exemple le choix de la paire de chaussures que l'on va porter durant la journée. D'autres par contre auront un impact plus important ; comme par exemple l'université où aller faire ses études.

Ces problèmes peuvent être classés en deux catégories. Certaines décisions – par exemple le choix d'une université – ne devront (généralement) être prises qu'une seule fois. Il n'est ici pas possible d'utiliser son expérience pour améliorer les futures décisions.

D'autres par contre se représenteront de manière séquentielle dans le temps. Le fait de se retrouver plusieurs fois devant le même choix permet d'acquérir de l'expérience en vue d'améliorer ses choix futurs. Par exemple, on peut comprendre rapidement que mettre des sandales lorsqu'il neige est un mauvais choix.

Dans cette thèse, nous allons étudier ce deuxième type de problème, c'est-à-dire les problèmes de sélection séquentielle dans un environnement incertain. Ce sont des problèmes d'*optimisation stochastique discrète* [80] qui peuvent être modélisés en associant à chaque alternative k une variable aléatoire $\mathbf{z}_k$ dont l'espérance μ_k est inconnue. Il s'ensuit que lorsqu'une alternative est testée, celle-ci produit un gain qui est fonction de la variable

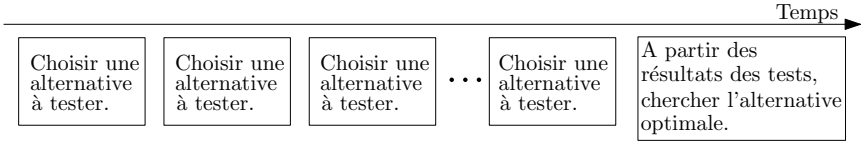


FIGURE 1.1 – Dans les problèmes de type *selecting the best*, il faut choisir séquentiellement les alternatives à tester de sorte qu'en fin de test, l'alternative optimale soit trouvée avec la plus grande probabilité.

aléatoire associée.

Selon l'objectif recherché, ces problèmes séquentiels peuvent encore être divisés en deux sous-catégories qui sont *les problèmes de la sélection de l'optimum avec un budget de tests limité* (*selecting the best*) et le *multi-armed bandit problem*. Nous les présentons ici brièvement.

1.1 La sélection de l'optimum avec un budget de tests limité

Le premier type de problème – *selecting the best* – est divisé en deux phases bien distinctes : une première phase durant laquelle un budget de H tests doit être utilisé de manière séquentielle sur les alternatives, suivie d'une deuxième phase qui a pour but de sélectionner – sur base des résultats obtenus durant les tests – l'alternative optimale (voir figure 1.1) ayant la plus grande espérance.

Il s'agit d'un problème d'*optimisation stochastique* bien connu dans la littérature scientifique, qui se retrouve notamment en simulation Monte Carlo [53] et en sélection de modèles dans l'apprentissage artificiel [59]. Il se rencontre dans de nombreuses situations pratiques, par exemple :

Un ingénieur en aéronautique peut avoir à développer une nouvelle aile d'avion. Les différentes configurations d'ailes peuvent être testées en soufflerie. Vu le coût excessif si de nombreuses configurations sont testées, il est courant d'utiliser des simulateurs informatiques durant les tests préliminaires [67]. Une seule simulation peut prendre plusieurs heures de calcul avant de proposer une estimation de la performance. Le résultat d'une simulation est généralement une observation d'un phénomène aléatoire. Une même configuration doit donc idéalement être testée plusieurs fois afin d'obtenir une estimation de sa performance moyenne. Le temps de calcul disponible peut ainsi rapidement devenir un frein à l'utilisation des techniques de simulation. L'ingénieur peut se fixer un budget de tests limité à H tests et doit alors appliquer de manière séquentielle une stratégie qui répartira les tests sur les diverses configurations. Cette stratégie est une fonction utilisant les performances déjà obtenues par les différentes configurations pour choisir la prochaine configuration à tester. Le but de la stratégie est qu'en fin de phase

FIGURE 1.2 – Dans les problèmes de type *bandit*, il faut choisir séquentiellement les alternatives à tester afin de maximiser la somme des gains collectés.

de test (phase durant laquelle les différentes configurations sont testées), la configuration ayant l'estimation de la performance moyenne la plus grande soit effectivement la configuration optimale.

Dans ce problème, il faut donc choisir de manière *séquentielle*, les configurations d'ailes à tester avec un *budget de tests* restreint. Puisque le résultat d'une simulation est incertain, les sélections des configurations à tester se font dans un *environnement aléatoire* et l'objectif est de trouver, *après les tests*, la configuration d'ailes *optimale*.

1.2 Le *multi-armed bandit problem*

L'objectif du deuxième type de problème – le *multi-armed bandit problem* (MABP) – est de sélectionner les différentes alternatives de manière séquentielle afin de maximiser la somme des gains collectés (voir figure 1.2).

C'est un problème d'*optimisation stochastique* bien connu et défini pour la première fois dans [71]. Le MABP est une formalisation du dilemme, dans un environnement incertain, entre des actions orientées vers l'exploitation et des actions orientées vers l'exploration [82]. Il se rencontre dans de nombreuses situations, par exemple :

En médecine, les praticiens sont souvent confrontés à des problèmes lorsque de nouveaux traitements sont proposés pour soigner une maladie. Une méthode d'évaluation consiste alors à diviser aléatoirement les patients en différents sous-groupes de taille égale et à appliquer ensuite le même traitement sur tous les patients d'un même sous-groupe. Cette technique donne de bons résultats d'un point de vue décisionnel mais il se peut que, très rapidement, un traitement s'avère supérieur aux autres. Il semblerait alors normal que les médecins privilégient les tests sur ce traitement ; mais cette méthode – peu flexible – ne permet pas ce type d'adaptation. Pour remédier à ce problème, une solution pourrait consister en une stratégie qui choisisse, sur base des résultats déjà obtenus, le traitement à appliquer aux nouveaux patients afin de maximiser le nombre de guérisons [44]. Ce critère a l'avantage de forcer la stratégie à trouver le meilleur traitement tout en optimisant le nombre de patients correctement traités.

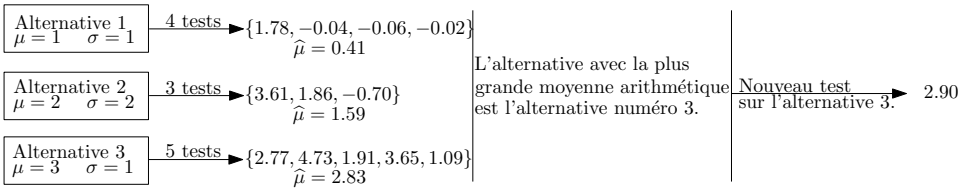


FIGURE 1.3 – Exemple d'utilisation d'une action de type glouton

Dans cet exemple, une action orientée exploitation consiste à utiliser le traitement qui semble – *pour l'instant* – donner les meilleures chances de guérison et une action orientée exploration consiste à utiliser un traitement insuffisamment testé.

Dans ce problème, il faut donc choisir de *manière séquentielle* les traitements à appliquer sur des patients. Puisque le résultat d'un traitement est incertain, les sélections des traitements à tester se font dans un *environnement aléatoire* et l'objectif est de *maximiser la somme* des patients guéris.

1.3 Formalisation du problème

Dans ces deux problèmes d'*optimisation stochastique*, il faut chaque fois tester de manière séquentielle une alternative afin de trouver – *le plus rapidement possible* – l'alternative optimale $[K]$ où $[K] = \arg\max_k \{\mu_k\}$.

Ces problèmes peuvent être modélisés via un ensemble de K alternatives associées à des variables aléatoires $\{\mathbf{z}_k\}_{k=1,\dots,K}$ de moyenne μ_k inconnue. Chaque fois qu'une alternative k est testée, celle-ci produit un gain et $\mathbf{z}_k^i$ est défini comme le gain après le i-ième test sur l'alternative k. Le choix de la prochaine alternative à tester se fait via une stratégie qui utilise les résultats des tests déjà obtenus. Le vecteur $\mathbf{Z}_k(n_k) = \left[\mathbf{z}_k^1, \mathbf{z}_k^2, \dots, \mathbf{z}_k^{n_k}\right]$ est un vecteur de n_k variables aléatoires indépendantes et identiquement distribuées qui contient les résultats obtenus après n_k tests sur l'alternative k.

Nous verrons que les stratégies de sélection de type glouton ont un rôle important dans la résolution des problèmes d'optimisation stochastique. Elles utilisent les observations collectées $\{\mathbf{Z}_1(n_1), \mathbf{Z}_2(n_2), \dots, \mathbf{Z}_K(n_K)\}$ pour calculer les moyennes arithmétiques $\{\widehat{\boldsymbol{\mu}}_1, \widehat{\boldsymbol{\mu}}_2, \dots, \widehat{\boldsymbol{\mu}}_K\}$ et sélectionnent l'alternative qui correspond à la variable aléatoire $\mathbf{z}_{[\widehat{\mathbf{K}}]}$ ayant la plus grande moyenne arithmétique c'est-à-dire $[\widehat{\mathbf{K}}] = \arg\max_k \{\widehat{\boldsymbol{\mu}}_k\}$.

La figure 1.3 donne un exemple d'action de type glouton qui comporte trois alternatives. Lorsqu'une alternative est testée, celle-ci engendre un gain. Comme nous supposons être dans un environnement incertain, les gains sont aléatoires et présentent des espérances et des écarts-types spécifiques à chaque alternative. Dans cet exemple, les moyennes des alternatives sont res-

pectivement de un, deux et trois, les écarts-types sont respectivement de un, deux et un et les alternatives ont été testées respectivement quatre fois, trois fois et cinq fois. Après avoir été testées, les alternatives ont généré une série de gains [1]. Les moyennes arithmétiques des gains des trois alternatives sont respectivement de 0.41, 1.59 et 2.83. Dans un problème de maximisation, une action de type glouton va sélectionner la troisième alternative. Après avoir été sélectionnée pour être testée, cette alternative donne un nouveau gain qui – dans cet exemple – vaut 2.90.

Dans le problème *selecting the best* il faut trouver l'alternative optimale $[K]$ avec un budget de tests fixe de taille H. Or, la nature aléatoire de l'environnement ne permet pas d'être certain que l'alternative sélectionnée soit l'optimum $[K]$. Les stratégies se focaliseront plutôt sur la quantité « *probabilité de faire une sélection correcte* » [53] qui sera notée par la suite $\Pr\{S_{[K]}\}$. Elle correspond à la probabilité que l'alternative $[\widehat{\mathbf{K}}]$ soit l'alternative optimale $[K]$; c'est-à-dire $\Pr\{S_{[K]}\} = \Pr\Big\{[\widehat{\mathbf{K}}] = [K]\Big\}$. Nous verrons qu'après la phase de test, la sélection de $[\widehat{\mathbf{K}}]$ correspond à une sélection de type *glouton*.

La fonction que l'on cherche à maximiser dans le cas du problème du *bandit* est ici différente puisqu'il s'agit de la somme des gains obtenus durant la phase de test. Il est à noter que dans ce cas, tester une alternative sous-optimale diminue le score final de la stratégie. Un juste compromis doit donc être trouvé entre les actions d'exploitation et les actions d'exploration. Une action orientée exploitation teste l'alternative se montrant – pour l'instant – optimale. Une action gloutonne fait donc partie de cette catégorie. Les actions orientées exploration testent plutôt les alternatives peu connues afin de maximiser les gains futurs obtenus via les prochaines actions d'exploitation. Une solution consiste donc à utiliser la quantité $\Pr\{S_{[K]}\}$ durant l'exploration car elle correspond à la probabilité que les actions d'exploitation gloutonne sélectionnent l'optimum $[K]$. Les actions d'exploration doivent donc chercher à maximiser cette quantité $\Pr\{S_{[K]}\}$.

D'un point de vue des techniques d'*aide à la décision*[72], il existe différentes manières de définir l'alternative optimale. Nous définissons, dans cette thèse, l'alternative optimale selon la quantité μ. Il est à noter qu'il est possible de définir l'alternative optimale selon une quantité plus complexe prenant par exemple également en compte la variance des variables associées aux alternatives.

1.4 Sélection séquentielle et sélection de modèles

L'apprentissage supervisé [85] est une technique qui consiste à construire une fonction prédictive d'un phénomène d'entrée/sortie sur base d'exemples

1. Dans cet exemple, les distributions des gains sont supposées suivre une loi de distribution normale.

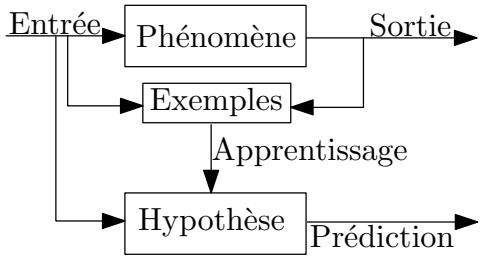


FIGURE 1.4 – Une série d'exemples d'un phénomène d'entrée/sortie est disponible. Ces exemples permettront de construire une hypothèse qui pourra ensuite être utilisée pour faire des prédictions sur de nouvelles entrées ne se trouvant pas dans l'ensemble d'exemples.

collectés à partir de ce phénomène. La fonction prédictive est appelée hypothèse. Le but est qu'une fois l'apprentissage terminé, il soit possible d'utiliser l'hypothèse pour faire des prédictions sur de nouvelles entrées ne se trouvant pas dans l'ensemble d'exemples (voir figure 1.4).

Une étape importante dans tous les processus d'apprentissage est la *sélection de modèles* [22]. Cette étape consiste à choisir, dans une série de K ensembles d'hypothèses, quel est l'ensemble d'hypothèses *optimal* qu'il faut utiliser pour sélectionner la fonction prédictive. La sélection de modèles peut être réalisée via des approches paramétriques ou non-paramétriques. Par exemple la technique de *Akaike Information Criterion* [3] est une approche paramétrique qui suppose que les ensembles d'hypothèses en compétition appartiennent à la famille des hypothèses linéaires. L'autre approche, non-paramétrique, ne fait pas de supposition quant aux ensembles d'hypothèses en compétition. Nous allons utiliser, dans le cadre de cette thèse, la technique non-paramétrique de sélection de modèles par *cross-validation* [81].

La sélection de modèles par *cross-validation* consiste à tester un grand nombre de fois chaque ensemble d'hypothèses et une estimation de la performance d'un ensemble d'hypothèses est obtenue après chaque test. C'est sur base des résultats des tests qu'un ensemble d'hypothèses optimal est sélectionné. Cette technique donne de bons résultats d'un point de vue décisionnel mais demande de faire un grand nombre de tests sur chacune des K alternatives.

Afin d'accélérer la procédure de sélection de modèles par *cross-validation*, le problème de sélection peut être considéré comme un problème de type *selecting the best* ayant un budget de tests fixé à H.

Il faut alors choisir, de manière *séquentielle*, les ensembles d'hypothèses à tester avec un *budget de tests* restreint. Puisque le résultat d'un test est incertain, les sélections des ensembles d'hypothèses à tester se font dans un *environnement aléatoire* et l'objectif est de trouver, *après les tests*, l'ensemble d'hypothèses *optimal*.

1.4.1 L'application des méthodes d'apprentissage artificiel en anesthésie

Cette thèse a été financée par un projet de la région Wallonne de type First Europe Objectif 1 numéro EP1A320501R059F / 415717, intitulé « *Data mining prédictif en anesthésie intraveineuse informatisée en vue de certification* » . Ce projet a été réalisé en collaboration avec le service d'anesthésie de l'Hôpital Erasme [2] et la société Mexys SA [3].

Lors d'une intervention chirurgicale, les anesthésistes de l'hôpital Erasme utilisent, pour piloter les différentes seringues, une application appelée ITB (*Infusion Toolbox*) [23]. Cette application enregistre un grand nombre de paramètres du patient ainsi que les décisions prises par l'anesthésiste (ex. le niveau de débit des drogues). Le sujet du projet First Europe Objectif 1 est l'application des méthodes d'apprentissage artificiel dans l'analyse des pratiques d'anesthésie en milieu hospitalier en utilisant les informations sauvegardées par ITB.

L'objectif est de concevoir une fonction prédictive afin de développer un système qui aide l'anesthésiste à choisir les concentrations de drogues idéales pour maintenir le patient dans un état d'inconscience désiré. Dans ce contexte, la sélection de modèles joue un rôle très important afin de construire une hypothèse ayant de bonnes capacités de prédiction. Ce projet First a permis d'appliquer les techniques développées dans cette thèse sur des données réelles provenant du milieu médical.

1.5 Les principales contributions de cette thèse

Cette thèse propose comme contributions de nouveaux algorithmes pour les problèmes « *selecting the best* » et « *bandit* » . Ces algorithmes sont tous basés sur une nouvelle quantité que nous proposons, appelée « *gain espéré d'une action gloutonne* » (noté μ_g). Cette quantité est construite sur la notion de « probabilité de faire une sélection correcte » (PCS) bien connue en simulation Monte Carlo.

- *Le gain espéré d'une action gloutonne.*
  - Comme nous sommes dans un environnement aléatoire, une action gloutonne ne renverra pas toujours la même valeur pour un nombre déterminé de tests sur les diverses alternatives. Nous avons introduit et défini de manière analytique la quantité μ_g qui est le gain moyen

2. L'Hôpital Erasme est l'hôpital académique de l'Université Libre de Bruxelles (Belgique).

3. La société Mexys SA – basée à Mons (Belgique) – est une société spécialisée dans la conception et la réalisation de produits liés à l'informatique médicale. Plus particulièrement, Mexys SA a développé un système d'acquisition automatique des données dans le domaine de l'anesthésie. Plus d'informations peuvent être obtenues sur leur site internet `http://www.mexys.com/`

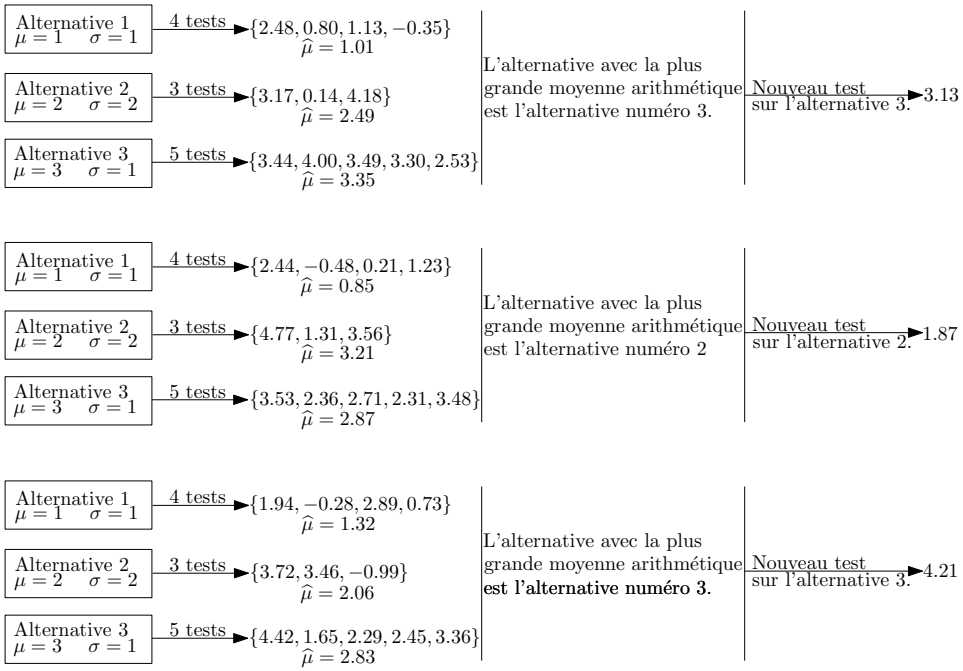


FIGURE 1.5 – Exemple où une action gloutonne est exécutée trois fois sur le même problème. Bien que le nombre de tests réalisés sur chaque alternative reste constant, le résultat obtenu est – après une action gloutonne – chaque fois différent (3.13, 1.87 et 4.21). La quantité « gain espéré d'une action gloutonne » est l'espérance des résultats obtenus après une action gloutonne.

obtenu après une série de sélections gloutonnes où le nombre de tests réalisés sur les différentes alternatives reste fixe (voir figure 1.5).
- Nous avons réalisé une étude analytique et empirique de l'évolution de μ_g lorsque le nombre de tests sur les différentes alternatives augmente. Lors de cette étude, nous avons identifié des situations où réaliser des tests supplémentaires sur les alternatives pouvait réduire les valeurs futures de μ_g.
- Pour calculer μ_g, la définition analytique de cette quantité utilise les moyennes μ_k et les écarts-types σ_k des alternatives. Comme ces valeurs sont supposées être inconnues, différents estimateurs de μ_g ont été proposés et comparés expérimentalement.

– *Les algorithmes pour les problèmes de type selecting the best.*
Trois algorithmes ont été proposés pour les problèmes de type *selecting the best*. Ils ont été comparés avec des méthodes faisant partie de l'état de l'art sur des problèmes créés de manière synthétique et sur des problèmes de sélection de modèles. Ces algorithmes sont : « optiK2 » , « *PCSsearch* » et « exploreMaxμ_g » .
  - L'algorithme optiK2 est spécifique pour le cas où seules deux al-

ternatives sont en compétition. A chaque étape, optiK2 sélectionne l'alternative qui maximise l'augmentation de la PCS à l'étape suivante.
  - L'algorithme *PCSsearch* cherche à répartir les tests sur les différentes alternatives de manière à explicitement faire augmenter la PCS lorsque plus de deux alternatives sont en compétition.
  - L'étude de l'évolution de μ_g a identifié différentes situations où tester une alternative peut faire diminuer la valeur de μ_g. L'algorithme exploreMaxμ_g utilise différentes *heuristiques* afin d'éviter de telles situations et de toujours faire augmenter μ_g.
- *Les algorithmes pour les problèmes de type bandit.*
  Deux algorithmes ont été proposés pour les problèmes de type *bandit*. Ils ont été comparés avec d'autres stratégies faisant partie de l'état de l'art. Ces algorithmes sont : ϵ-PCSgreedy et DP-*greedy*.
  - Lors des actions d'exploration, ϵ-PCSgreedy utilise l'algorithme *PCSsearch* afin de faire explicitement augmenter la PCS et ceci pour maximiser les futurs gains des actions d'exploitation.
  - L'algorithme DP-*greedy* est construit dans un cadre de programmation dynamique afin de mesurer l'impact d'une action sur les valeurs futures de μ_g. Ceci permet à la stratégie de s'adapter en fonction de cette évolution.
- *L'application des méthodes d'apprentissage en anesthésie.*
  - Les algorithmes que nous proposons pour résoudre les problèmes de type *selecting the best* seront appliqués sur une tâche de sélection de variables avec des données réelles d'anesthésie.
  - Un système d'aide à la décision en anesthésie a également été développé et est déjà validé en conditions réelles.
  - Deux patients différents peuvent réagir de manière très variée lorsque – dans le même contexte – une même quantité de drogue leur est injectée. Afin de permettre à l'hypothèse de s'adapter à cette variabilité interindividuelle, nous avons proposé une technique qui consiste à utiliser les premiers instants de l'intervention pour ajuster le modèle à la sensibilité du patient aux drogues.

1.6 Structure de la thèse

Les deux chapitres suivants concernent l'état de l'art des méthodes qui résolvent les problèmes de sélection de modèles en apprentissage artificiel (chapitre 2) et les problèmes de sélection séquentielle dans un environnement aléatoire (chapitre 3). Les contributions de cette thèse commencent au chapitre 4. Ce chapitre définit la quantité du *gain espéré d'une action gloutonne* μ_g, étudie ensuite son évolution lorsque le nombre de tests sur les alternatives change et, pour terminer, propose différentes méthodes pour es-

timer μ_g. Le chapitre 5 utilise la quantité μ_g pour proposer trois algorithmes qui résolvent le problème *selecting the best*. Les performances de ces algorithmes seront évaluées sur différents cas tests créés de manière synthétique et sur des problèmes de sélection de modèles construits à partir de données réelles. Le chapitre 6 concerne les contributions dans le domaine du *bandit*. Deux nouveaux algorithmes – basés sur la quantité μ_g – seront proposés et testés sur des problèmes synthétiques et réels. Dans le chapitre 7, des méthodes d'apprentissage artificiel seront utilisées dans le cadre d'un programme d'aide à la décision appliqué en anesthésie. Ce programme permet d'estimer le niveau d'inconscience du patient dans une fenêtre temporelle de dix minutes après la modification du niveau des drogues. Ceci permettra également de tester – sur des données réelles provenant du milieu médical – les méthodes de sélection de modèles proposées dans le chapitre 5. Enfin, le dernier chapitre contient les conclusions de la thèse et les perspectives futures.

1.7 Publications

Les travaux présentés dans cette thèse ont fait l'objet de différentes publications dont voici la liste dans un ordre chronologique inverse :

- Caelen O., Cailloux O., Ghoundiwal D., Miranda A. A., Barvais L., Bontempi G. Real-time prediction of an anesthetic monitor index using machine learning. *Submitted*, 2009.
- Caelen O., Bontempi G. A dynamic programming strategy to balance exploration and exploitation in the bandit problem. *Submitted*, 2009.
- Miranda A. A., Caelen O., Bontempi G. Machine Learning for Automated Polyp Detection in Computed Tomography Colonography. *Biomedical Image Analysis and Machine Learning Technologies : Applications and Techniques*. Editors : Fabio Gonzalez and Eduardo Romero. Publisher : IGI Global, Hershey, USA. Book Chapter accepted for publication, 2009.
- Caelen O., Bontempi G. On the evolution of the expected gain of a greedy action in the bandit problem. *ULB Technical Report* Number 589, 2008.
- Caelen O., Bontempi G. Improving the exploration strategy in bandit algorithms. Proceedings of *Learning and Intelligent OptimizatioN LION II* , Lecture Notes in Computer Science, Springer, 56-68, 2007.
- Caelen O., Bontempi G., Barvais L. Machine learning techniques for decision support in anesthesia. Proceedings of the *11th Conference on Artificial Intelligence in Medicine (AIME2007)* , Lecture Notes in Computer Science, Springer, 165-169, 2007.
- Van Rompaey N., Engelman E., Caelen O., Verleije A., Ickx B. Influence of Entropy Monitoring on Drug Consumption during and after

Long Lasting TCI Anesthesia (abstract). *Euroanaesthesia 2007 Meeting*, June 2007, Munich

- Caelen O., Bontempi G., Coussaert E., Barvais L., Clément F. Machine learning techniques to enable closed-loop control in anesthesia. Proceedings of the *19th IEEE International Symposium on Computer-Based Medical Systems (CBMS2006)*, 696-701, 2006.
- Caelen O., Engelman E., Schmartz D., Bontempi G., Barvais L. Preliminary Results of Data Mining in BIS guided Propofol-Remifentanil TCI Anaesthesia (abstract). *Euroanaesthesia 2006 Meeting*, June 2006, Madrid.
- Caelen O., Bontempi G., Clement F., Coussaert E., Barvais L. Simulation assessment of a closed-loop controller designed by machine learning techniques (abstract). *9 th Eurosiva Meeting*, June 2006, Madrid.
- Bejjani G., Caelen O., Bontempi G., Perrin L., Barvais L. Retrospective comparison of manual versus semi-automated propofol-remifentanil TCI Anaesthesia (abstract). *9 th Eurosiva Meeting*, June 2006, Madrid.
- Caelen O., Bontempi G. How to allocate a restricted budget of leave-one-out assessments for effective model selection in machine learning : a comparison of state-of-the-art techniques. Proceedings of the *17th Belgian-Dutch Conference on Artificial Intelligence*, 51-58, 2005.
- Meyer P.E., Caelen O., Bontempi G. Speeding up feature selection by using an information theoretic bound. Proceedings of the *17th Belgian-Dutch Conference on Artificial Intelligence*, 166-173, 2005.
- Bontempi G., Caelen O., Pierret S., Goffaux C. On the use of supervised learning techniques to speed up the design of aeronautics components. Proceedings of The *WSEAS Transactions on Systems*, 3098-3103, 2005.

1.8 Notation

Les symboles en police grasse sont utilisés pour désigner les variables aléatoires alors que leurs réalisations se présentent en police normale. Il est toujours possible, se basant sur le contexte, de savoir si une variable est ou non aléatoire. Nous adoptons cette convention uniquement pour des raisons de lisibilité.

Une police *inclinée* est utilisée sur les termes anglophones.

ERM	*Empirical Risk Minimisation.*
MISE	*Mean Integrated Squared Error.*
MSE	*Mean Squared Error.*
MABP	*Multi-armed bandit problem.*
SB	*Selecting the best.*
$\widehat{e}$	L'accent circonflexe sur une variable indique qu'il s'agit d'un estimateur.
$\mathbf{e}$	La police grasse indique qu'il s'agit d'une variable aléatoire.
$\mathbb{E}[\cdot]$	Espérance d'une variable aléatoire.
$F(\cdot)$	Répartition de probabilités d'une variable aléatoire.
$\Pr\{\cdot\}$	Probabilité.
$f(\cdot)$	Fonction cible. On suppose que $\mathbf{y} = f(x) + \mathbf{w}$.
$\mathbf{D}_N$	Ensemble composé de N exemples d'apprentissage. $\mathbf{D}_N = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N\}$.
N	Nombre d'exemples dans l'ensemble $\mathbf{D}_N$.
α	Caractéristiques d'une hypothèse.
$h(\cdot, \alpha)$	Hypothèse.
G_N	Erreur de généralisation.
$\widehat{G}_N^{loo}$	Estimation de l'erreur de généralisation par *leave-one-out*.
K	Nombre d'alternatives disponibles.
k	Indice utilisé pour indexer les alternatives.
H	Nombre de rounds disponibles.
l	Indice utilisé pour indiquer le numéro du round actuel.
n_k	Nombre de fois que $\mathbf{z}_k$ a été testé.
μ	Moyenne ou espérance mathématique.
$\widehat{\boldsymbol{\mu}}$	Moyenne arithmétique.
σ	Ecart-type.
$\widehat{\boldsymbol{\sigma}}$	Estimation de l'écart-type.

$[K] = \arg\max \mu_k$	Indice de la variable ayant la plus grande espérance. De manière générale, $[i]$ représente l'indice de la variable qui a la i-ième plus petite moyenne.
$[\widehat{\mathbf{K}}] = \arg\max \widehat{\boldsymbol{\mu}}_k$	Indice de la variable ayant la plus grande moyenne arithmétique.
$\widehat{\mathbf{k}}$	Indice d'une alternative sélectionnée par une stratégie pour être testée.
$\Pr\{S_k\}$	Probabilité de sélectionner l'alternative k où $\Pr\{S_{[K]}\}$ est la probabilité de faire une sélection correcte.
μ_g	Gain espéré d'une action gloutonne.
μ_r	Gain espéré d'une action aléatoire uniforme.
π	Stratégie.

Chapitre 2

Méthodes de sélection de modèles en apprentissage supervisé

Dans un problème d'apprentissage supervisé, un superviseur génère un ensemble d'apprentissage à partir d'un phénomène aléatoire. Un algorithme d'apprentissage automatique utilise ensuite cet ensemble d'apprentissage pour construire une hypothèse du phénomène qui pourra alors servir à effectuer des prédictions sur le phénomène. Ce chapitre introduit les concepts de la théorie de l'apprentissage supervisé qui seront utiles dans la suite de cette thèse. Une attention particulière sera donnée au problème de sélection de modèles. La section 2.1 formalise le problème d'apprentissage supervisé et définit les notations. Le compromis estimation-approximation est introduit dans la section 2.2 et montre qu'il faut réaliser, lors d'un apprentissage supervisé, une sélection de modèles via un algorithme d'identification structurelle. L'identification structurelle est introduite à la section 2.3 et la section 2.4 traite des méthodes de sélection de modèles par minimisation de l'erreur de généralisation. La section 2.5 introduit les méthodes de sélection de variables utilisées dans le chapitre 7. Ce chapitre se termine par la section 2.6 qui présente quelques familles d'ensembles d'hypothèses.

2.1 Formulation du problème d'apprentissage supervisé

Le problème d'apprentissage supervisé peut être décrit d'un point de vue statistique par les éléments suivants [85] (voir figure 2.1) :

– un *générateur* de vecteurs aléatoires $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^d$ générés de manière indépendante à partir d'une fonction de répartition de probabilités $F_{\mathbf{x}}(x)$,

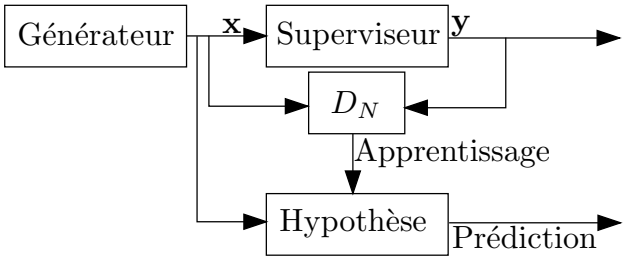


FIGURE 2.1 – Dans un problème d'apprentissage supervisé, un générateur crée des variables d'entrée et un superviseur transforme celles-ci en variables de sortie. Nous supposons qu'un ensemble d'exemples d'entrée/sortie est collecté dans D_N où D_N est une réalisation de la variable aléatoire $\mathbf{D}_N$. Sur base de cet ensemble d'exemples, un algorithme d'apprentissage construit une hypothèse pouvant ensuite être utilisée pour faire des prédictions sur la sortie du superviseur pour de nouvelles entrées.

- un *superviseur* qui renvoie pour toute valeur d'entrée x une valeur de sortie $y \in \mathcal{Y}$ en fonction d'une répartition de probabilités conditionnelles $F_{\mathbf{y}}(y \mid x)$. Ceci inclut le cas $\mathbf{y} = f(x) + \mathbf{w}$, c'est-à-dire que le superviseur génère les valeurs de sortie via une *fonction cible* $f(\cdot)$ à laquelle un bruit $\mathbf{w}$ est ajouté et où le bruit $\mathbf{w}$ est généré à partir d'une fonction de répartition $F_{\mathbf{w}}(w)$. Nous supposons que l'espérance du bruit est nulle,
- un *ensemble d'apprentissage* $D_N = \{\langle x_1, y_1\rangle, \langle x_2, y_2\rangle, \ldots, \langle x_N, y_N\rangle\}$ composé de N paires $\langle x_i, y_i\rangle \in \mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ de variables indépendantes et identiquement distribuées selon une fonction de répartition de probabilités $F(\langle x, y\rangle) = F_{\mathbf{y}}(y \mid x) \cdot F_{\mathbf{x}}(x)$,
- un *algorithme d'apprentissage* qui, sur base d'une réalisation D_N de la variable aléatoire $\mathbf{D}_N$, peut « *construire* » un prédicteur de la fonction cible $f(\cdot)$ appelée hypothèse.

Dire que l'algorithme d'apprentissage peut « *construire* » une hypothèse signifie en réalité qu'il choisit cette dernière parmi un ensemble d'hypothèses. Le problème de l'apprentissage supervisé consiste donc à trouver dans un ensemble d'hypothèses et sur base d'un ensemble d'apprentissage D_N fixe, l'hypothèse optimale estimant le mieux la fonction cible $f(\cdot)$ inconnue.

Une hypothèse est notée $h(\cdot, \alpha)$ où α définit les caractéristiques de l'hypothèse. Comme une hypothèse est intégralement définie par ses caractéristiques, une hypothèse $h(\cdot, \alpha)$ sera parfois simplement dénotée dans les équations par α.

Λ est défini comme un ensemble de caractéristiques possibles (voir tableau 2.1). Comme les caractéristiques d'une hypothèse définissent intégralement cette dernière, nous dirons parfois par abus de langage que Λ est un ensemble d'hypothèses. C'est dans cet ensemble Λ que l'algorithme d'ap-

Indice	Familles d'hypothèses	Paramètres de l'hypothèse
1	Quadratique : $a_3 \cdot x^2 + a_2 \cdot x + a_1$	$a_1 = 7$ $a_2 = 3$ $a_3 = 1.4$
2	Quadratique : $a_3 \cdot x^2 + a_2 \cdot x + a_1$	$a_1 = 3.8$ $a_2 = 5.2$ $a_3 = 2$
3	Linéaire : $a_2 \cdot x + a_1$	$a_1 = 1.2$ $a_2 = 3.6$
4	Perceptron multicouche avec deux neurones cachés : $g(x \cdot a_1) \cdot a_3 + g(x \cdot a_2) \cdot a_4$ où $g(\cdot)$ est une fonction sigmoïdale [1]	$a_1 = 1.1$ $a_2 = 6.8$ $a_3 = 6.4$ $a_4 = 1.7$

TABLE 2.1 – Dans ce tableau, Λ contient seulement quatre caractéristiques d'hypothèses possibles. Il s'agit d'un exemple simplifié car Λ contient généralement une infinité d'éléments. Ces quatre éléments de Λ définissent les caractéristiques de quatre hypothèses : $h(x, \alpha{=}1) = 1.4 \cdot x^2 + 3 \cdot x + 7$, $h(x, \alpha = 2) = 2 \cdot x^2 + 5.2 \cdot x + 3.8$, $h(x, \alpha = 3) = 3.6 \cdot x + 1.2$ et $h(x, \alpha{=}4) = g(x \cdot 1.1) \cdot 6.4 + g(x \cdot 6.8) \cdot 1.7$. Une valeur de α définit donc intégralement une hypothèse.

prentissage choisit, sur base de l'ensemble D_N, l'hypothèse estimant le mieux la fonction cible.

Un algorithme d'apprentissage peut utiliser différents ensembles d'hypothèses pour désigner l'hypothèse optimale. Il est courant, en apprentissage supervisé, de présenter les différents ensembles d'hypothèses d'une manière imbriquée $\Lambda_1 \subset \Lambda_2 \subset \ldots \subset \Lambda_k \subset \ldots \subset \Lambda_K = \Lambda_*$ où Λ_* désigne l'ensemble contenant les caractéristiques de toutes les hypothèses possibles.

Dans le but de chercher l'hypothèse optimale estimant le mieux la fonction cible, la *fonction de coût* $\mathcal{C}(f(x), h(x, \alpha))$ est définie. Cette fonction mesure, pour un x donné, l'écart entre la fonction cible $f(\cdot)$ et l'hypothèse $h(\cdot, \alpha)$. Sur base de cette fonction de coût, la capacité d'approximation d'une hypothèse $h(\cdot, \alpha)$ est mesurée via la *fonction risque* R. La fonction risque R fait la moyenne, pour une hypothèse $h(\cdot, \alpha)$ donnée, de la fonction de coût

1. Une fonction sigmoïdale est définie comme une fonction dont l'expression est $g(x) = 1/(1 + e^{\lambda x})$ où λ est un paramètre. Son nom vient de sa courbe en forme de S.

sur le $\mathcal{X}-$ domaine

$$\begin{aligned} R(\alpha) &= \int_{\mathcal{X}} \mathcal{C}(f(x), h(x,\alpha))\ dF_{\mathbf{x}}(x) \\ &= \int_{\mathcal{X},\mathcal{Y}} \mathcal{C}(y, h(x,\alpha))\ dF_{\mathbf{y}}(y \mid x)\ dF_{\mathbf{x}}(x). \end{aligned} \tag{2.1}$$

L'hypothèse $h(\cdot, \alpha_+^k)$ avec $\alpha_+^k \in \Lambda_k$ est définie comme l'hypothèse dans Λ_k qui minimise la fonction risque

$$\alpha_+^k = \arg\min_{\alpha \in \Lambda_k} R(\alpha) \tag{2.2}$$

et l'hypothèse $h(\cdot, \alpha_*)$ avec $\alpha_* \in \Lambda_*$ est définie comme l'hypothèse dans Λ_* avec le plus petit risque fonctionnel possible

$$\alpha_* = \arg\min_{\alpha \in \Lambda_*} R(\alpha). \tag{2.3}$$

Le but de l'apprentissage supervisé est donc de trouver l'hypothèse $h(\cdot, \alpha_*)$ avec $\alpha_* \in \Lambda_*$ lorsque les fonctions de répartition $F_{\mathbf{x}}(x)$ et $F_{\mathbf{y}}(y \mid x)$ sont inconnues et lorsque seule une réalisation D_N de la variable aléatoire $\mathbf{D}_N$ est disponible.

Le *risque empirique* R_{emp} mesure l'erreur de prédiction moyenne d'une hypothèse sur l'ensemble d'apprentissage D_N

$$R_{emp}(\alpha) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{C}\left(y_i, h(x_i, \alpha)\right). \tag{2.4}$$

Dans un problème d'apprentissage supervisé, l'algorithme d'apprentissage est composé d'un *algorithme d'identification paramétrique* $\mathcal{I}^P$ fondé sur le principe *ERM* (*Empirical Risk Minimization*) [85] qui consiste à chercher, dans l'ensemble Λ, l'hypothèse $h(\cdot, \alpha_N)$ minimisant le risque empirique

$$\alpha_N = \arg\min_{\alpha \in \Lambda} R_{emp}(\alpha). \tag{2.5}$$

La section 2.1.2 à la page 19 donne un exemple de résolution du problème de minimisation du risque empirique.

Cette formulation du problème d'apprentissage supervisé est très générale et permet d'étudier différents problèmes classiques en statistique. La prochaine section étudie le problème de régression [29]. Vapnik, dans les sections 1.3 et 1.6 de [85], définit également, dans le même cadre formel, les problèmes de classification [31] et d'estimation de densité de probabilité [75], que nous ne considérerons pas dans ce travail.

2.1.1 Le problème de régression

Supposons que pour toute valeur d'entrée x un superviseur renvoie une valeur de sortie $y \in \mathcal{Y} \subset \mathbb{R}$ où y est la réalisation d'une probabilité conditionnelle $F_{\mathbf{y}}(y \mid x)$. L'estimation de la dépendance stochastique entre $\mathbf{x}$ et $\mathbf{y}$ sur base des N exemples requiert l'estimation de la distribution conditionnelle $F_{\mathbf{y}}(y \mid x)$. Dans un problème de régression [29], on ne cherche pas à estimer la distribution $F_{\mathbf{y}}(y \mid x)$ mais uniquement l'espérance de $\mathbf{y}$, notée $f(\cdot)$

$$f(x) = \mathbb{E}[\mathbf{y} \mid x] = \int_{\mathcal{Y}} y dF_{\mathbf{y}}(y \mid x).$$

La fonction $f(\cdot)$ est appelée fonction cible. Le problème de régression consiste donc à estimer cette fonction cible $f(\cdot)$ quand la fonction de répartition $F_{\mathbf{y}}(y \mid x)$ est inconnue mais que nous disposons d'un ensemble d'apprentissage de N exemples générés à partir de la fonction de répartition $F(\langle x, y \rangle)$. Si la fonction de coût est égale, pour un x donné, au carré de la différence entre l'hypothèse et la fonction cible

$$\mathcal{C}(f(x), h(x, \alpha)) = (f(x) - h(x, \alpha))^2, \tag{2.6}$$

alors nous pouvons montrer que le minimum de la fonction risque

$$R(\alpha) = \int_{\mathcal{X}} (f(x) - h(x, \alpha))^2 \, dF_{\mathbf{x}}(x)$$

est atteint par une hypothèse dans Λ égale à la fonction cible si celle-ci fait partie de la classe d'hypothèses Λ.

2.1.2 Exemple du problème d'identification paramétrique

Cette section a pour but d'illustrer, par un exemple, le problème de minimisation du risque empirique dans le cas de la régression. On considère le problème suivant :

- les entrées sont créées via une distribution uniforme dans l'intervalle [2] $[0, \pi]$,
- le superviseur génère les sorties en fonction d'une distribution normale de moyenne égale à $\sin(x^2) - \cos(x)$ et d'écart type égal à 1,
- on dispose d'un ensemble d'apprentissage D_N composé de $N = 50$ paires $\langle x_i, y_i \rangle$ d'entrée/sortie (voir figure 2.2). D_N est une réalisation de la variable aléatoire $\mathbf{D}_N$.

Sur la seule base des cinquante exemples dans D_N, et sans aucune autre connaissance, nous allons chercher à construire une hypothèse pouvant être utilisée par la suite pour effectuer des prédictions.

2. Dans cet exemple, un espace d'entrée $\mathcal{X}$ à une dimension a été choisi. Ce choix a été fait uniquement pour des raisons de lisibilité des figures. Les notions introduites se généralisent aisément à des problèmes à plus d'une variable d'entrée.

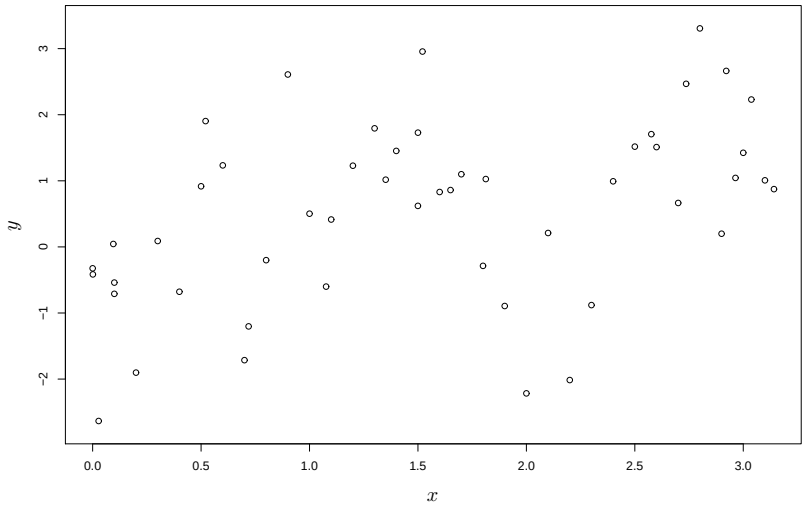


FIGURE 2.2 – Ensemble d'apprentissage pour le problème de régression composé de $N = 50$ paires d'entrée/sortie. Les entrées sont générées d'une manière aléatoire en fonction d'une distribution uniforme dans l'intervalle $[0, \pi]$. Les sorties sont générées via la fonction cible $f(x) = \sin(x^2) - \cos(x)$ à laquelle est ajouté un bruit Gaussien d'espérance nulle et d'écart type un.

Nous choisissons un ensemble Λ très simple correspondant à l'ensemble de toutes les fonctions linéaires passant par l'origine. L'ensemble Λ est donc composé des hypothèses $h(x, \alpha) = a \cdot x$ où a est le paramètre de coefficient de proportionnalité à optimiser. Avec une fonction de coût $\mathcal{C}$ quadratique, le risque empirique est

$$R_{emp}(\alpha) = \sum_{i=1}^{50} (y_i - h(x_i, \alpha))^2 .$$

Le rôle de l'identification paramétrique $\mathcal{I}^P$ est de trouver dans l'ensemble Λ les caractéristiques de l'hypothèse qui minimisent le risque empirique et donc, dans cet exemple, de trouver le paramètre a optimal [3]. La figure 2.3 montre l'évolution du risque empirique en fonction du paramètre a. Nous pouvons voir sur cette figure que le risque empirique est minimum lorsque a est égal à 0.42. La figure 2.4 montre l'hypothèse $h(\cdot, \alpha_N)$ où α_N correspond aux caractéristiques d'une fonction linéaire passant par l'origine et ayant comme coefficient de proportionnalité $a = 0.42$. Cette hypothèse $h(\cdot, \alpha_N)$,

3. Dans le cas de fonctions linéaires, la minimisation du risque empirique se fait normalement via la technique du *least-squares* (voir annexe B.1 à la page 169). Avec cette méthode, il n'est pas utile de calculer R_{emp} pour toutes les valeurs possibles de a. Nous le faisons ici uniquement à titre d'exemple.

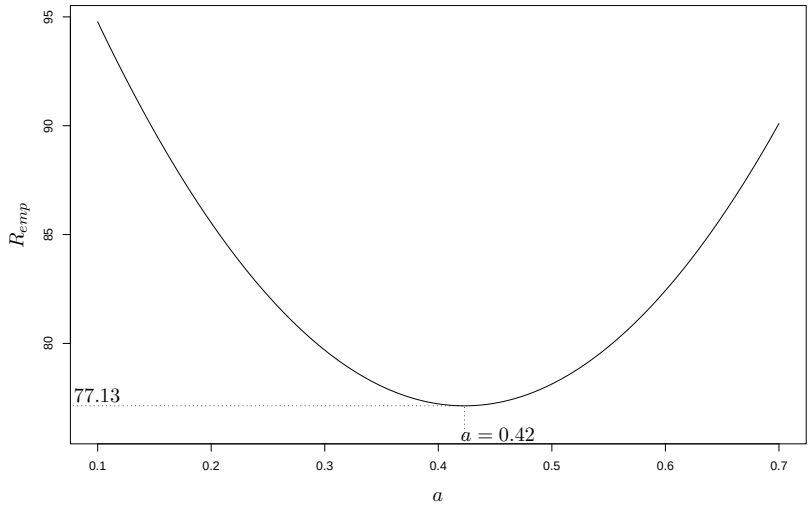


FIGURE 2.3 – Evolution du risque empirique lorsque le paramètre a évolue entre 0.1 et 0.7. La valeur $a = 0.42$ est la valeur du paramètre qui minimise le risque empirique.

choisie via le principe ERM qui minimise le risque empirique, pourra ensuite être utilisée pour faire des prédictions sur de nouvelles entrées.

Pour l'instant, qualitativement, nous pouvons voir que l'hypothèse sélectionnée réalise encore de mauvaises prédictions. L'hypothèse $h(\cdot, \alpha_N)$ est l'hypothèse – dans l'ensemble d'hypothèses linéaires passant par l'origine – qui modélise le mieux l'ensemble d'apprentissage D_N. Le fait que $h(\cdot, \alpha_N)$ estime mal la fonction cible $f(\cdot)$ est donc dû à Λ. Pour améliorer la précision de l'hypothèse sélectionnée par ERM, il faut alors que l'ensemble Λ contienne plus d'hypothèses afin de trouver une hypothèse qui estime mieux D_N. Mais choisir un ensemble Λ trop riche peut également donner de mauvais résultats car l'hypothèse sélectionnée par ERM dans un tel cas modéliserait si bien l'ensemble D_N qu'il risquerait de devenir trop sensible au bruit. Ceci constitue le sujet de la prochaine section.

2.2 Le compromis estimation-approximation

Supposons qu'un algorithme d'apprentissage dispose d'un ensemble d'apprentissage D_N qui est une réalisation de la variable aléatoire $\mathbf{D}_N$ de distribution de probabilités inconnue. Nous avons vu, à la page 18, que l'algorithme d'apprentissage doit chercher l'hypothèse optimale $h(\cdot, \alpha_*)$ ayant le plus petit risque fonctionnel possible. Pour cela, il utilise le principe ERM (voir équation (2.5) à la page 18) et choisit, dans un ensemble Λ, l'hypothèse

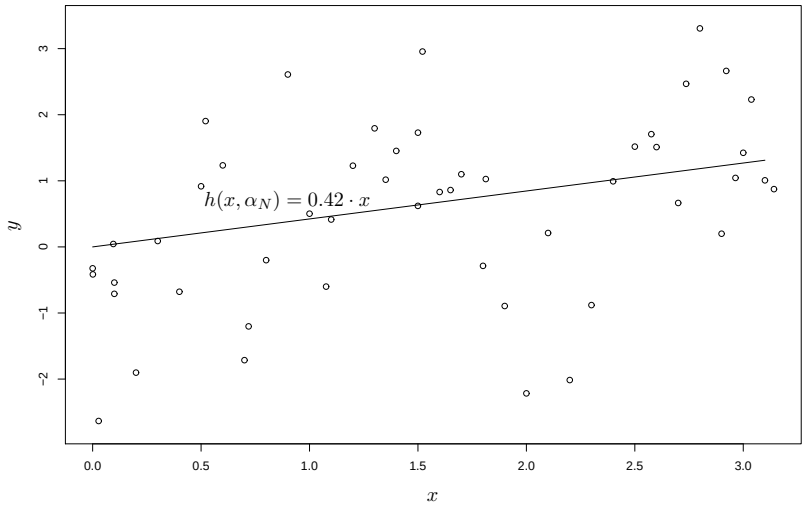


FIGURE 2.4 – L'hypothèse $h(\cdot, \alpha_N)$ avec $\alpha_N \in \Lambda$ minimisant le risque empirique, où Λ contient les caractéristiques de toutes les hypothèses linéaires passant par l'origine.

$h(\cdot, \alpha_N)$ minimisant le risque empirique où $\alpha_N \in \Lambda$.

Le compromis estimation-approximation (voir section 2.2.1 de [27]) exprime l'effet de différents facteurs influant sur la différence entre le risque fonctionnel de l'hypothèse choisie par l'algorithme d'apprentissage $R(\alpha_N)$ et le risque fonctionnel de l'hypothèse optimale $R(\alpha_*)$. Nous pouvons écrire l'équation suivante (voir figure 2.5)

$$\begin{aligned} R(\alpha_N) - R(\alpha_*) &= (R(\alpha_N) - R(\alpha_+)) \ + \ (R(\alpha_+) - R(\alpha_*)) \\ &= \mathrm{Err}_{estim}(\alpha_N) + \mathrm{Err}_{approx} \end{aligned}$$

où α_+ est défini par l'équation (2.2) à la page 18. Il est courant de définir le premier terme comme l'erreur d'estimation et le deuxième terme comme l'erreur d'approximation. Nous allons maintenant étudier l'origine de ces deux types d'erreur et nous verrons qu'elles sont diamétralement opposées de telle sorte que diminuer une erreur peut faire augmenter l'autre. Pour minimiser $R(\alpha_N) - R(\alpha_*)$, il a y donc un juste compromis à trouver entre l'erreur d'estimation et l'erreur d'approximation.

L'erreur d'estimation $\mathrm{Err}_{estim}(\alpha_N)$ est due au fait que l'algorithme d'apprentissage ne fournit en général pas l'hypothèse optimale $h(\cdot, \alpha_+)$ dans Λ mais calcule $h(\cdot, \alpha_N)$ avec $\alpha_N \in \Lambda$ sur base d'un échantillon de taille N. Plus l'ensemble Λ est riche, plus $h(\cdot, \alpha_N)$ risque d'être éloigné de $h(\cdot, \alpha_+)$ et plus le nombre N augmente, plus $h(\cdot, \alpha_N)$ a des chances de se rapprocher de $h(\cdot, \alpha_+)$. Pour un D_N fixé, l'erreur d'estimation augmente donc généralement avec la richesse de l'ensemble Λ.

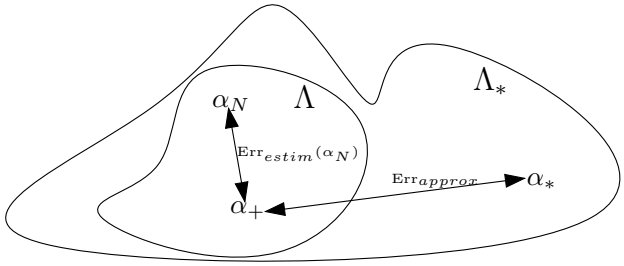


FIGURE 2.5 – La différence entre le risque fonctionnel de l'hypothèse choisie par le principe de minimisation du risque empirique $R(\alpha_N)$ et le risque fonctionnel de l'hypothèse optimale $R(\alpha_*)$ est égale à la somme des erreurs d'estimation $\mathrm{Err}_{estim}(\alpha_N)$ et d'approximation Err_{approx}.

L'erreur d'approximation Err_{approx} est liée à la capacité d'approximation de la classe Λ et ne dépend pas de l'ensemble d'apprentissage D_N. Cette erreur est due au fait que l'ensemble Λ ne contient pas le modèle optimal. Pour diminuer l'erreur d'approximation il faut augmenter le nombre d'hypothèses dans l'ensemble Λ de telle sorte que la différence entre $R(\alpha_+)$ et $R(\alpha_*)$ soit réduite.

Il faut donc trouver un compromis sur la taille de l'ensemble Λ. Si l'ensemble Λ est trop vaste, $R(\alpha_+)$ a des chances d'être proche de $R(\alpha_*)$ mais l'erreur d'estimation sera probablement grande. En revanche si l'ensemble Λ est trop réduit, c'est l'erreur d'approximation qui risque cette fois d'être très élevée.

Pour choisir l'hypothèse optimale, un algorithme d'apprentissage peut utiliser différents ensembles d'hypothèses. Pour trouver la taille optimale de l'ensemble d'hypothèses, nous avons vu qu'il est courant de présenter les ensembles d'hypothèses en une collection de sous-ensembles emboîtés $\Lambda_1 \subset \Lambda_2 \subset \ldots \subset \Lambda_k \subset \ldots \subset \Lambda_K$. Donc plus l'indice k est grand, plus l'espace d'hypothèses est riche. Une procédure de recherche est alors exécutée pour trouver l'ensemble Λ optimal. Cette procédure de recherche sur les ensembles d'hypothèses est exécutée par un algorithme d'*identification structurelle* réalisant une *sélection de modèles* [22].

2.2.1 Illustration du compromis estimation-approximation

Cette section a pour but de montrer, par un exemple, qu'il existe un compromis à trouver sur la taille de l'ensemble des hypothèses. Nous allons montrer qu'appliquer une identification paramétrique sur un ensemble trop ou pas assez riche engendre un mauvais estimateur de la fonction cible. Nous considérons le même problème de régression que celui de la section 2.1.2 où l'ensemble d'apprentissage D_N est composé de $N = 50$ paires $\langle x_i, y_i \rangle$ d'entrée/sortie (voir figure 2.2 à la page 20). L'ensemble D_N est une réali-

sation de la variable aléatoire $\mathbf{D}_N$ de distribution de probabilités inconnue.

Nous considérons les vingt-trois classes d'hypothèses suivantes :

- Λ_0 est l'ensemble des hypothèses constantes tel que $h(x, \alpha^0) = a_0$ si $\alpha^0 \in \Lambda_0$,
- Λ_1 est l'ensemble des hypothèses linéaires tel que $h(x, \alpha^1) = a_1 \cdot x + a_0$ si $\alpha^1 \in \Lambda_1$,
- Λ_2 est l'ensemble des hypothèses quadratiques tel que $h(x, \alpha^2) = a_2 \cdot x^2 + a_1 \cdot x + a_0$ si $\alpha^2 \in \Lambda_2$,
- Λ_3 est l'ensemble des hypothèses polynomiales du troisième degré tel que $h(x, \alpha^3) = a_3 \cdot x^3 + a_2 \cdot x^2 + a_1 \cdot x + a_0$ si $\alpha^3 \in \Lambda_3$,
- ...
- Λ_{22} est l'ensemble des hypothèses polynomiales du vingt-deuxième degré tel que $h(x, \alpha^{22}) = a_{22} \cdot x^{22} + \ldots + a_2 \cdot x^2 + a_1 \cdot x + a_0$ si $\alpha^{22} \in \Lambda_{22}$.

Il est évident que les différents ensembles sont imbriqués $\Lambda_0 \subset \Lambda_1 \subset \ldots \subset \Lambda_k \subset \ldots \subset \Lambda_{22}$. Par exemple, si le paramètre a_2 est fixé à zéro dans Λ_2 alors l'ensemble Λ_2 devient équivalent à l'ensemble Λ_1. Dans cet exemple, nous constatons donc que plus l'indice k d'une classe d'hypothèses est grand, plus la classe d'hypothèses est riche.

$h(\cdot, \alpha_N^k)$ est défini comme l'hypothèse choisie dans Λ_k via l'algorithme d'identification paramétrique (voir équation (2.5) à la page 18). Vu que dans cet exemple la fonction cible est connue et vaut $\sin(x^2) - \cos(x)$, il est possible de calculer le risque R (voir équation (2.1) à la page 18) d'une hypothèse $h(\cdot, \alpha_N^k)$ où, pour rappel (voir page 19), les variables d'entrée du problème sont créées d'une manière uniforme dans le domaine $\mathcal{X} = [0, \pi]$

$$R(\alpha_N^k) = \int_0^\pi \left(\sin(x^2) - \cos(x) - h(x, \alpha_N^k)\right)^2 \, dF_{\mathbf{x}}(x). \tag{2.7}$$

Il est à noter qu'en général, la fonction cible et la distribution du générateur de $\mathbf{x}$ sont inconnues et que la fonction R n'est donc pas accessible.

La figure 2.6 montre, pour un ensemble d'apprentissage donné en figure 2.2 à la page 20, l'évolution du risque R en fonction de la richesse de l'ensemble Λ. Elle montre qu'une identification paramétrique $\mathcal{I}^P$ sur une classe Λ trop petite (comme Λ_0) construit une mauvaise hypothèse. Dans un ensemble Λ_0, ne contenant que des hypothèses constantes, l'erreur d'approximation sera très grande car même l'hypothèse optimale $h(\cdot, \alpha_+^0)$ avec $\alpha_+^0 \in \Lambda_0$ sera toujours très éloignée de la fonction cible $\sin(x^2) - \cos(x)$. Nous dirons qu'une hypothèse construite à partir d'un ensemble Λ trop petit fait de l'*under-fitting*.

Faire une identification paramétrique $\mathcal{I}^P$ sur une classe Λ trop grande (comme Λ_{22}) construit une mauvaise hypothèse car l'erreur d'estimation sera très grande. Nous dirons qu'une hypothèse construite à partir d'un ensemble Λ trop riche fait de l'*over-fitting*.

Dans cet exemple, l'hypothèse $h(\cdot, \alpha_N^6)$ est l'hypothèse qui minimise la fonction risque R. L'ensemble Λ_6 est l'ensemble d'hypothèses obtenant le

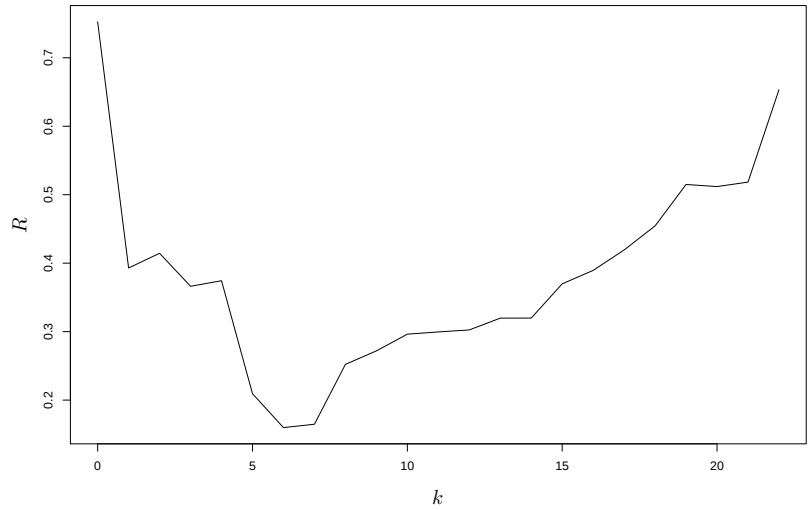


FIGURE 2.6 – Evolution de la fonction risque R de l'hypothèse $h(\cdot, \alpha_N)$ lorsque la classe d'hypothèses devient de plus en plus riche.

meilleur compromis entre l'erreur d'estimation et l'erreur d'approximation.

La figure 2.7 montre les hypothèses $h(\cdot, \alpha_N^0)$, $h(\cdot, \alpha_N^6)$ et $h(\cdot, \alpha_N^{22})$. Nous constatons que l'hypothèse $h(\cdot, \alpha_N^0)$ modélise mal les exemples de D_N car elle est trop « simple » . La classe Λ_{22} est à l'inverse tellement riche que $h(\cdot, \alpha_N^{22})$ a commencé à modéliser le bruit ce qui fait osciller l'hypothèse. L'hypothèse $h(\cdot, \alpha_N^6)$ semble être une bonne hypothèse et pourra être utilisée pour faire des prédictions sur de nouvelles entrées non présentes dans D_N (voir figure 2.8).

Dans cet exemple, nous avons choisi Λ_6 comme la meilleure classe d'hypothèses. Chercher la meilleure classe d'hypothèses, c'est faire une *sélection de modèles* réalisée par un algorithme d'*identification structurelle*. Il est à noter que dans cet exemple, nous avons « triché » dans le sens qu'en conditions réelles, les fonctions de répartition de probabilités $F_{\mathbf{y}}(y \mid x)$ et $F_{\mathbf{x}}(x)$ sont inconnues et qu'il faut donc réaliser l'identification structurelle sans connaître la fonction risque R (voir équation (2.7)). Les sections 2.3 et 2.4 traiteront du problème de sélection de modèles dans le cas réel où la fonction R est inaccessible.

2.3 L'identification structurelle

Pour trouver le juste compromis entre l'erreur d'estimation et l'erreur d'approximation, un algorithme d'identification structurelle est utilisé. Cet algorithme d'identification structurelle opère une sélection de modèles. La

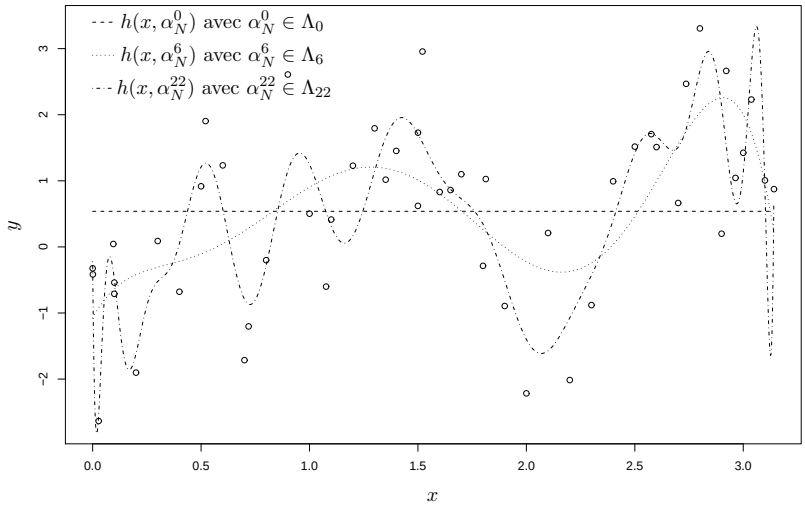


FIGURE 2.7 – Exemple de trois hypothèses sélectionnées par l'identification paramétrique dans les ensembles Λ_0, Λ_6 et Λ_{22}.

démarche consiste à considérer un ensemble d'hypothèses et à le décomposer d'une manière imbriquée $\Lambda_1 \subset \Lambda_2 \subset \ldots \subset \Lambda_k \subset \ldots$. Il est à noter que la fonction cible $f(\cdot)$ peut ou non être incluse dans l'un de ces ensembles d'hypothèses. Sur base d'un ensemble d'apprentissage D_N fixé, l'hypothèse $h(\cdot, \alpha_N^k)$ est définie comme l'hypothèse sélectionnée dans Λ_k via le principe ERM consistant à chercher dans l'ensemble Λ_k l'hypothèse minimisant le risque empirique (voir équation (2.5) à la page 18). Sur base d'un ensemble D_N, le problème de sélection de modèles consiste à trouver la classe d'hypothèses $\Lambda_{\bar{k}}$ telle que le risque fonctionnel $R(\alpha_N^{\bar{k}})$ soit réduit au minimum

$$\bar{k} = \arg\min_k R(\alpha_N^k)$$

où $\alpha_N^{\bar{k}}$ a été choisi via le principe ERM dans l'ensemble $\Lambda_{\bar{k}}$. Deux types d'approches, pour résoudre le problème de sélection de modèles, sont présentés dans cette section :

– le principe de minimisation du risque structurel (*structural risk minimization* (SRM)) proposé par Vapnik [84, 85]. Dans cette approche, nous considérons que l'ensemble D_N est fixé et que la variable aléatoire $\mathbf{D}_N$ n'est pas accessible. Nous cherchons des bornes sur le risque fonctionnel $R(\cdot)$ ne dépendant pas de la distribution de probabilités de $\mathbf{D}_N$. Pour cela, Vapnik a défini la VC-dimension $\mathfrak{h}$ d'un ensemble Λ mesurant la capacité de généralisation de l'ensemble d'hypothèses Λ. Le risque empirique $R_{emp}(\alpha_N^k)$ est ensuite pénalisé par un terme $\mathcal{E}(\cdot, \cdot, \cdot)$ fonction de la VC-dimension $\mathfrak{h}_k$, du nombre d'exemples d'apprentis-

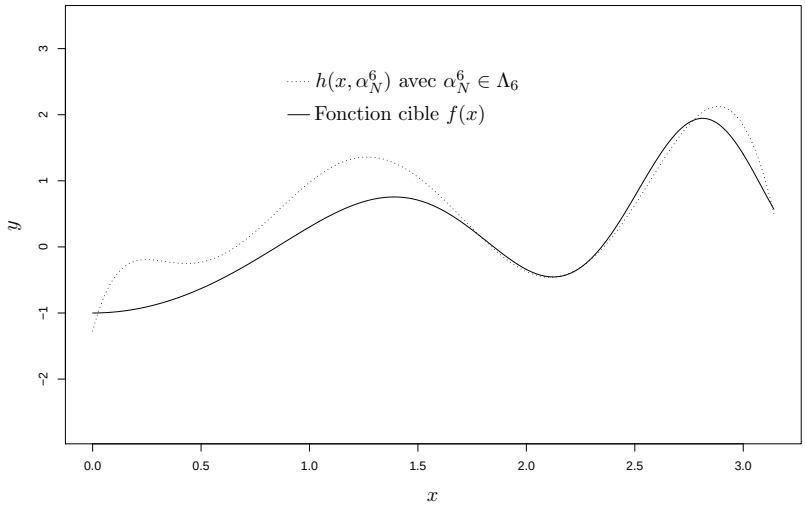


FIGURE 2.8 – Sur cette figure, la fonction cible $f(x) = \sin(x^2) - \cos(x)$ et l'hypothèse $h(x, \alpha_N^6)$ sont affichées. Nous constatons que $h(x, \alpha_N^6)$ peut être utilisé pour faire des prédictions sur $f(x)$. Il y a deux raisons pour expliquer qu'il existe encore une différence entre $f(x)$ et $h(x, \alpha_N^6)$. La première raison est que $h(x, \alpha_N^6)$ a été choisi dans un ensemble Λ_6 ne contenant pas la fonction cible (erreur d'approximation) et la deuxième raison est que $h(x, \alpha_N^6)$ est une approximation faite sur $N = 50$ exemples bruités (erreur d'estimation).

sage N et d'une confiance $1 - \delta$. Nous obtenons alors la borne

$$R(\alpha_N^k) \leq R_{emp}(\alpha_N^k) + \mathcal{E}(\mathfrak{h}_k, N, \delta) = SRM(k)$$

ne dépendant pas de la distribution de probabilités de $\mathbf{D}_N$. Le principe SRM choisit ensuite l'hypothèse $h(\cdot, \alpha_N^{\bar{k}})$ où

$$\bar{k} = \arg\min_k SRM(k).$$

– le principe de validation par apprentissage multiple. Ici, nous considérons l'ensemble D_N comme une réalisation de la variable aléatoire $\mathbf{D}_N$ et le risque empirique devient une fonction de $\mathbf{D}_N$

$$\mathbf{R}_{emp}(\alpha) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{C}\left(\mathbf{y}_i, h(\mathbf{x}_i, \alpha)\right).$$

Comme $\mathbf{R}_{emp}(\cdot)$ dépend de $\mathbf{D}_N$, $\mathbf{R}_{emp}(\cdot)$ est une variable aléatoire. Le fait que $\mathbf{R}_{emp}(\cdot)$ devient aléatoire a pour conséquence que le risque

fonctionnel $R(\boldsymbol{\alpha}_N)$ devient également aléatoire. L'*erreur de généralisation* G_N est la moyenne, sur l'ensemble d'apprentissage $\mathbf{D}_N$ et pour une classe Λ, du risque fonctionnel des hypothèses minimisant le risque empirique

$$G_N(\Lambda) = \mathbb{E}[R(\boldsymbol{\alpha}_N) \mid \Lambda] \,.$$

Le principe est alors de choisir l'hypothèse $h(\cdot, \alpha_N^{\bar{k}})$ où ici

$$\bar{k} = \arg\min_k G_N(\Lambda_k).$$

Dans cette thèse, c'est cette deuxième approche qui est étudiée.

2.4 Sélection de modèles par minimisation de l'erreur de généralisation

Le problème de sélection de modèles consiste à chercher la classe d'hypothèses sur laquelle l'application du principe ERM (voir équation (2.5) à la page 18) détermine l'hypothèse ayant le plus petit risque fonctionnel possible.

Le principe de validation par apprentissage multiple est étudié dans cette section. Ici, l'ensemble d'apprentissage D_N est considéré comme une réalisation d'une variable aléatoire $\mathbf{D}_N$. Comme l'ensemble d'apprentissage est aléatoire, l'hypothèse $h(\cdot, \boldsymbol{\alpha}_N)$ et son risque fonctionnel $R(\boldsymbol{\alpha}_N)$ deviennent également aléatoires. Pour en obtenir l'espérance, le risque fonctionnel doit donc être calculé sur une série d'apprentissages effectués sur différentes réalisations de l'ensemble d'apprentissage. L'erreur de généralisation G_N est définie comme l'espérance du risque fonctionnel sur les différentes réalisations de l'ensemble d'apprentissage. Le principe de sélection de modèles par apprentissages multiples consiste ensuite à sélectionner l'ensemble d'hypothèses qui minimise l'erreur de généralisation.

2.4.1 L'erreur de généralisation

L'*erreur de généralisation* G_N est définie comme la moyenne, sur l'ensemble d'apprentissage $\mathbf{D}_N$ et pour une classe d'hypothèses Λ, du risque fonctionnel $R(\cdot)$ des hypothèses qui minimisent le risque empirique (voir figure 2.9)

$$\begin{aligned} G_N(\Lambda) &= \mathbb{E}[R(\boldsymbol{\alpha}_N) \mid \Lambda] \\ &= \int_{\mathcal{Z}^N, \mathcal{X}, \mathcal{Y}} \mathcal{C}(y, h(x, \mathcal{I}^P(\Lambda, D_N))) \;\; dF_{\mathbf{y}}(y \mid x) \; dF_{\mathbf{x}}(x) \; dF_{\mathbf{D}_N}(D_N) \end{aligned} \tag{2.8}$$

où $\mathcal{I}^P$ est un algorithme d'identification paramétrique appliquant le principe ERM. Pour une fonction de coût quadratique, l'erreur de généralisation G_N

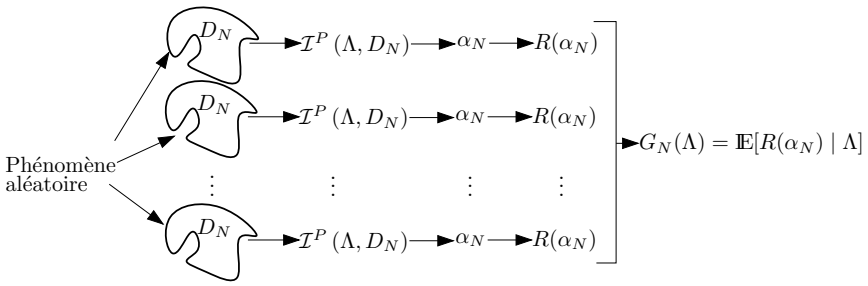


FIGURE 2.9 – Liens entre le risque fonctionnel R et l'erreur de généralisation G_N. L'erreur de généralisation est égale à la moyenne de la fonction risque de l'hypothèse $h(\cdot, \alpha_N)$ sur l'ensemble des D_N possibles.

devient le *mean integrated squared error* (MISE)

$$\text{MISE} = \int_{\mathcal{Z}^N, \mathcal{X}, \mathcal{Y}} (y - h(x, \mathcal{I}^P(\Lambda, D_N)))^2 \; dF_{\mathbf{y}}(y \mid x) \; dF_{\mathbf{x}}(x) \; dF_{\mathbf{D}_N}(D_N). \tag{2.9}$$

En fonction de la variable aléatoire $\mathbf{D}_N$, l'espérance de la fonction de coût $\mathcal{C}$ pour une entrée x donnée est

$$\overline{C}_N(x) = \int_{\mathcal{Z}^N, \mathcal{Y}} \mathcal{C}(y - h(x, \mathcal{I}^P(\Lambda, D_N))) \; dF_{\mathbf{y}}(y \mid x) \; dF_{\mathbf{D}_N}(D_N).$$

Lorsque la fonction de coût $\mathcal{C}$ est quadratique, cette quantité est généralement appelée *mean squared error* (MSE). Le MSE est utilisé dans la section suivante pour définir le compromis biais-variance.

Sur base de $\overline{C}_N(\cdot)$, l'erreur de généralisation est égale à

$$\begin{aligned} G_N(\Lambda) &= \mathbb{E}\big[\overline{C}_N(\mathbf{x})\big] \\ &= \int_{\mathcal{X}} \overline{C}_N(x) \; dF_{\mathbf{x}}(x) \end{aligned}$$

et donc

$$\text{MISE}(\Lambda) = \int_{\mathcal{X}} \text{MSE}(x) \; dF_{\mathbf{x}}(x). \tag{2.10}$$

2.4.2 Le compromis biais-variance

L'utilisation d'une fonction de coût $\mathcal{C}$ de type quadratique permet de décomposer le MSE en trois termes [38]

$$\begin{aligned}
\text{MSE} &= \mathbb{E}\Big[(\mathbf{y} - h(x, \boldsymbol{\alpha}_N))^2\Big] \\
&= \mathbb{E}\Big[(\mathbf{y} - \mathbb{E}[\mathbf{y} \mid x])^2\Big] + \mathbb{E}\Big[(h(x, \boldsymbol{\alpha}_N) - \mathbb{E}[\mathbf{y} \mid x])^2\Big] \\
&= \mathbb{E}\Big[(\mathbf{y} - \mathbb{E}[\mathbf{y} \mid x])^2\Big] + (\mathbb{E}[h(x, \boldsymbol{\alpha}_N)] - \mathbb{E}[\mathbf{y} \mid x])^2 \\
&\quad + \mathbb{E}\Big[(h(x, \boldsymbol{\alpha}_N) - \mathbb{E}[h(x, \boldsymbol{\alpha}_N)])^2\Big] \\
&= \text{bruit} + \text{carré du biais} + \text{variance}.
\end{aligned}$$

Cette décomposition du MSE est appelée la décomposition biais-variance. Le MSE peut donc s'écrire comme suit

$$MSE(x) = \sigma^2_{\mathbf{w}}(x) + \text{biais}(x)^2 + \text{variance}(x). \tag{2.11}$$

La quantité $\sigma^2_{\mathbf{w}}$ mesure la variance du bruit, donc son intensité (l'espérance du bruit est supposée être nulle). Il s'agit d'une information ne concernant que l'ensemble d'apprentissage.

Le biais mesure la différence entre la moyenne de la valeur de sortie par l'hypothèse en x et la valeur de la fonction à modéliser $f(x)$. Le biais est une information ne concernant que la classe d'hypothèses et la fonction cible. Afin de minimiser le biais, il faut utiliser un ensemble Λ plus riche, capable de modéliser des fonctions cibles plus complexes.

Enfin, la variance mesure la sensibilité des hypothèses sélectionnées via ERM lorsque l'ensemble d'apprentissage varie. Une trop grande variance est signe que Λ risque d'être trop sensible au bruit et donc qu'une classe Λ trop riche est utilisée. La variance est une information qui ne concerne que la classe d'hypothèses.

En combinant les équations (2.10) et (2.11), nous obtenons

$$\text{MISE} = \int_{\mathcal{X}} \sigma^2_{\mathbf{w}}(x)\ dF_{\mathbf{x}}(x) + \int_{\mathcal{X}} \text{biais}(x)^2\ dF_{\mathbf{x}}(x) + \int_{\mathcal{X}} \text{variance}(x)\ dF_{\mathbf{x}}(x). \tag{2.12}$$

La décomposition biais-variance peut donc également être appliquée sur le MISE.

Pour minimiser l'erreur de généralisation, un compromis entre le biais et la variance doit être réalisé. Ce compromis est appelé communément le dilemme biais/variance (*the bias/variance dilemma*). Ce type de compromis a été évoqué à la section 2.2 pour la minimisation du risque fonctionnel. Appliquer le principe ERM sur une classe d'hypothèses très riche produit des hypothèses ayant un petit biais mais les hypothèses produites

par l'algorithme d'apprentissage auront une grande variance, c'est une situation d'*over-fitting*. Si l'ensemble d'hypothèses Λ est très petit, alors les hypothèses sélectionnées dans Λ, via le principe ERM, ne varieront pas énormément lorsque l'ensemble d'apprentissage D_N change en fonction de la variable aléatoire $\mathbf{D}_N$. La variance de l'ensemble Λ est donc dans ce cas petite. Mais avoir un petit ensemble d'hypothèses Λ risque de produire des hypothèses biaisées si la fonction cible est complexe. Si l'ensemble Λ est trop petit, il s'agit d'une situation d'*under-fitting*.

2.4.2.1 Illustration du compromis biais-variance

Dans cette section, nous montrons par un exemple, qu'un compromis doit être trouvé quant à la richesse de l'ensemble d'hypothèses afin de minimiser l'erreur de généralisation. Le même problème d'apprentissage que celui de la section 2.2.1 à la page 23 est repris hormis que cette fois, l'ensemble d'apprentissage est aléatoire et qu'une série d'apprentissages sur différentes réalisations de $\mathbf{D}_N$ sont exécutés. Cette série d'apprentissages multiples permet de calculer le biais et la variance. Dans cette section, trois ensembles d'hypothèses sont testés (voir page 24) :

- l'ensemble des hypothèses constantes Λ_0,
- l'ensemble des hypothèses polynomiales du sixième degré Λ_6,
- l'ensemble des hypothèses polynomiales du vingt-deuxième degré Λ_{22}.

Cinq mille réalisations de la variable aléatoire $\mathbf{D}_N$ sont générées pour obtenir $\left\{D_N^1, D_N^2, \ldots, D_N^{5000}\right\}$. Une identification paramétrique est réalisée sur chaque ensemble d'apprentissage. Pour rappel, l'erreur de généralisation est fonction de la variance et du carré du biais (voir équation (2.12)). Les cinq mille réalisations sont utilisées pour estimer ce biais et cette variance. La figure 2.10 contient l'évolution du biais et de la variance sur le domaine d'entrée $\mathcal{X}$. Pour une question de lisibilité, les axes des graphiques ne correspondent pas au carré du biais et à la variance mais au biais et à l'écart-type.

La classe Λ_0 est une petite classe qui se traduit par une variance relativement faible mais cette petite variance se paie par un grand biais. L'autre cas extrême est le cas où la classe d'hypothèses est trop étendue. Par exemple, pour Λ_{22} le biais est petit mais la variance est grande. La classe Λ_6 est l'ensemble présentant le meilleur compromis entre le biais et la variance.

2.4.3 L'estimation de l'erreur de généralisation

Le calcul de l'erreur de généralisation G_N suppose que les distributions $F_{\mathbf{y}}(y \mid x)$ et $F_{\mathbf{x}}(x)$ soient connues (voir équation (2.8) à la page 28), ce qui n'est pas le cas en conditions réelles. Le problème consiste donc à estimer l'erreur de généralisation sur base d'un ensemble d'apprentissage D_N de N exemples $\langle x, y\rangle \in \mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ où D_N est une réalisation de la variable aléatoire $\mathbf{D}_N$ distribuée selon la fonction de répartition de probabi-

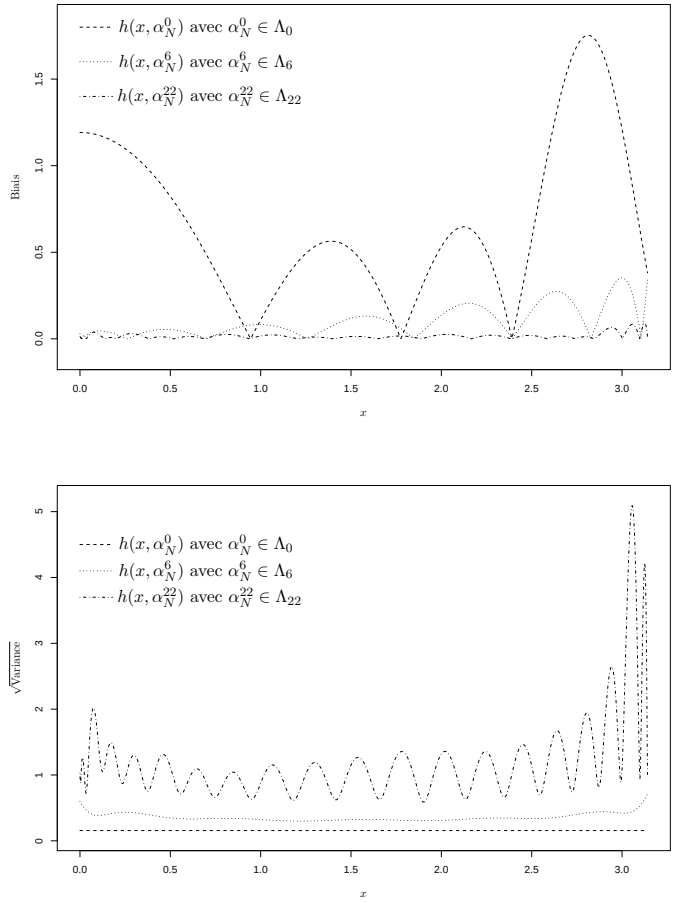


FIGURE 2.10 – Le biais et la variance sont estimés sur cinq mille ensembles d'apprentissage indépendants et identiquement distribués selon $\mathbf{D}_N$. Trois classes d'hypothèses sont comparées : Λ_0, Λ_6 et Λ_{22}.

lités $F_{\mathbf{y}}(y \mid x) \cdot F_{\mathbf{x}}(x)$. L'erreur de généralisation peut être estimée soit de manière analytique soit par rééchantillonnage.

Le *final prediction error* [2] et l'*Akaike's information criterion* [3] sont deux exemples de techniques analytiques pour estimer l'erreur de généralisation. Ils supposent que les exemples de l'ensemble d'apprentissage ont été créés via un générateur ayant une fonction cible linéaire.

L'estimation par rééchantillonnage ne fait, quant à elle, aucune supposition sur la distribution $\mathbf{D}_N$ ayant généré l'ensemble d'apprentissage D_N. Il existe différentes méthodes de rééchantillonnage, par exemple : la méthode de *cross-validation* [81] et la méthode du *bootstrap* [33].

La méthode du *bootstrap* considère D_N comme le générateur d'ensembles d'apprentissage (voir figure 2.11). A partir du générateur D_N, une série d'en-

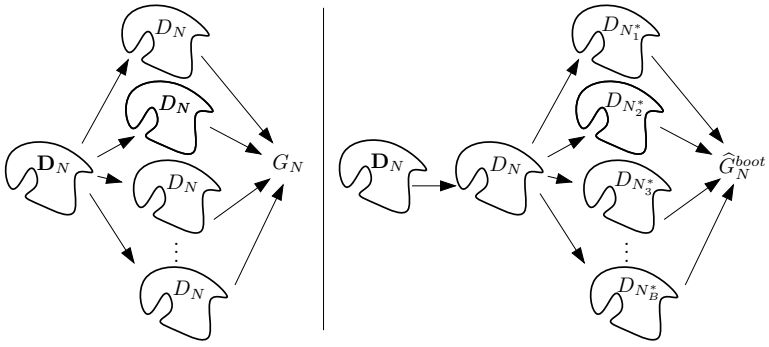


FIGURE 2.11 – A gauche, la variable aléatoire $\mathbf{D}_N$ est accessible permettant de créer une infinité de réalisations de $\mathbf{D}_N$ et il est donc possible de calculer G_N. Dans la réalité, la variable aléatoire $\mathbf{D}_N$ n'est pas accessible et nous ne disposons que d'une réalisation de $\mathbf{D}_N$. A droite, l'ensemble D_N est considéré comme le générateur d'ensembles d'apprentissage. L'ensemble D_N est échantillonné B fois *avec remise*, permettant de calculer l'estimateur de G_N par *bootstrap*.

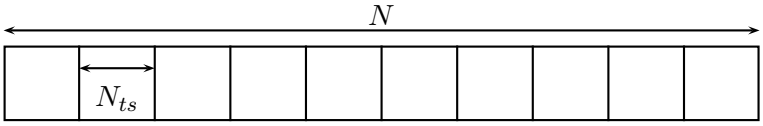


FIGURE 2.12 – L'ensemble D_N est décomposé en $\mathcal{K}$ parties. A chaque étape, $N_{tr} = N - N_{ts}$ exemples sont utilisés pour l'apprentissage et les N_{ts} autres exemples d'apprentissage sont utilisés pour l'évaluation de l'erreur.

sembles d'apprentissage $\left\{D_{N_1^*}, \ldots, D_{N_b^*}, \ldots, D_{N_B^*}\right\}$ est créée par échantillonnage dans D_N *avec remise*. Cette série d'ensembles d'apprentissage est utilisée par le *bootstrap* pour estimer G_N. Cet estimateur possède une variance relativement faible [34] mais cette variance limitée se paie par un fort biais [37]. Le *bootstrap*632 (voir section 17.7 de [33]) et le *bootstrap*632+ [37] sont des méthodes essayant de diminuer ce biais.

Dans cette étude, nous utilisons les estimateurs par *cross-validation*. L'idée de la *cross-validation* [81] est de découper l'ensemble d'apprentissage $\mathcal{K}$ fois en deux parties ; un sous-ensemble $D_{N_{tr}}$ et un sous-ensemble $D_{N_{ts}}$ (voir figure 2.12). A chaque itération, un sous-ensemble $D_{N_{tr}^i}$, composé de N_{tr} éléments, est utilisé lors de l'apprentissage pour construire l'hypothèse $h(\cdot, \alpha_{N_{tr}^i})$. L'autre sous-ensemble $D_{N_{ts}^i}$ est composé de N_{ts} exemples et est utilisé, quant à lui, pour calculer l'erreur de prédiction de $h(\cdot, \alpha_{N_{tr}^i})$

$$\widehat{R}(\alpha_{N_{tr}^i}) = \frac{1}{\left|D_{N_{ts}^i}\right|} \sum_{\langle x_j, y_j \rangle \in D_{N_{ts}^i}} \mathcal{C}(y_j, h(x_j, \alpha_{N_{tr}^i})).$$

A la fin de la procédure de *cross-validation*, une moyenne arithmétique est réalisée et renvoyée comme estimation de l'erreur de généralisation de l'hypothèse

$$\widehat{G}_N^{cv} = \frac{1}{\mathcal{K}} \sum_{i=1}^{\mathcal{K}} \widehat{R}(\alpha_{N_{tr}^i}). \tag{2.13}$$

Il est à noter que cette procédure demande d'exécuter $\mathcal{K}$ identifications paramétriques et N prédictions.

Le *leave-one-out cross-validation* est le cas extrême de la *cross-validation* dans lequel $\mathcal{K}$ est égal au nombre N d'exemples disponibles. L'ensemble $D_{(i)}$ est défini comme l'ensemble d'apprentissage D_N duquel l'exemple $\langle x_i, y_i \rangle$ a été retiré

$$D_{(i)} = D_N \setminus \{\langle x_i, y_i \rangle\} \tag{2.14}$$

et $h(\cdot, \alpha_{N(i)})$ est l'hypothèse correspondante qui minimise R_{emp}. L'erreur de généralisation calculée par le *leave-one-out cross-validation* est donc

$$\begin{aligned} \widehat{G}_N^{loo} &= \frac{1}{N} \sum_{i=1}^{N} \mathcal{C}(y_i, h(x_i, \alpha_{N(i)})) \\ &= \frac{1}{N} \sum_{i=1}^{N} e_{loo}^i \end{aligned} \tag{2.15}$$

où e_{loo}^i est l'erreur de *leave-one-out* calculée sur l'exemple $\langle x_i, y_i \rangle$.

Il est à souligner que, comme D_N est une réalisation de la variable aléatoire $\mathbf{D}_N$, la quantité $\widehat{G}_N^{loo}$ est également la réalisation d'une variable aléatoire $\widehat{\mathbf{G}}_N^{loo}$. L'erreur de généralisation estimée par *leave-one-out* est un estimateur « presque » non-biaisé [28], c'est-à-dire

$$\mathbb{E}_{\mathbf{D}_N}\left[\widehat{\mathbf{G}}_N^{loo}\right] = G_{N-1}.$$

En contrepartie, pour calculer $\widehat{G}_N^{loo}$, il est nécessaire de réaliser N identifications paramétriques ce qui – dans certains cas – demande un temps de calcul trop important pour être utilisé en pratique.

2.5 La sélection de variables

Pour obtenir des hypothèses ayant de bonnes capacités de prédiction, il est courant de faire appel à des techniques de réduction de la dimensionnalité du problème via des méthodes de sélection de variables [56, 55] (en anglais *feature selection*). Un trop grand nombre de variables d'entrée conduit fréquemment à des problèmes d'*over-fitting*.

Pour illustrer ce problème d'*over-fitting*, supposons une tâche de modélisation avec $d = 4$ variables d'entrée. Soit $\mathcal{M}1$ un masque sur les variables

Algorithme 1 Algorithme SFS

1: Fixer la classe d'hypothèses Λ qui sera utilisée.
2: $\mathcal{M}_0 \leftarrow \phi$.
3: $\mathcal{V} \leftarrow \{1, 2, \ldots, d\}$ où d est le nombre total de variables d'entrée.
4: $i \leftarrow 1$.
5: **tant que** $|\mathcal{V}| \neq 0$ **faire**
6: **pour tout** v dans $\mathcal{V}$ **faire**
7: $v_* \leftarrow$ choisir par *leave-one-out* la variable v avec laquelle le masque $\mathcal{M}_{i-1} \cup v$ a la meilleure erreur de généralisation G_N.
8: **fin pour**
9: $\mathcal{V} \leftarrow \mathcal{V} \setminus \{v_*\}$.
10: $\mathcal{M}_i \leftarrow \mathcal{M}_{i-1} \cup \{v_*\}$.
11: $i \leftarrow i + 1$.
12: **fin tant que**
13: A la fin de l'algorithme : $\mathcal{M}_1$ contient la meilleure variable, $\mathcal{M}_2$ contient les deux meilleures variables, etc. Et le masque ayant la plus petite erreur de généralisation est choisi.

d'entrée qui vaut $[1, 2, 4]$. Ceci signifie que toutes les variables sont utilisées à l'exception de la troisième. Soit $\mathcal{M}2$ qui vaut $[1, 4]$ ce qui signifie que seules la première et la dernière variable sont utilisées. Si Λ est une classe d'hypothèses quelconque et que $\mathcal{M}1(\Lambda)$ signifie que le masque $\mathcal{M}1$ est appliqué sur les entrées des hypothèses de Λ, alors il est évident que $\mathcal{M}2(\Lambda) \subset \mathcal{M}1(\Lambda)$ car pour obtenir $\mathcal{M}2(\Lambda)$ à partir de $\mathcal{M}1(\Lambda)$, il suffit dans $\mathcal{M}1(\Lambda)$ de ne pas considérer la deuxième variable.

L'utilisation d'un nombre trop élevé de variables d'entrée risque d'engendrer un ensemble d'hypothèses trop grand, ce qui conduit à un problème d'*over-fitting*. A l'inverse, l'utilisation de trop peu de variables risque de conduire à un problème d'*under-fitting*. Il y a donc à nouveau un juste compromis à trouver quant à la richesse de l'ensemble (voir section (2.2) à la page 21).

Les méthodes par filtrage [56] et les méthodes *warper* [55] sont deux exemples de techniques qui cherchent à réduire le nombre de variables. Dans cette thèse, ce sont les méthodes *warper* appelées *sequential forward selection* (SFS) [1] qui sont utilisées et c'est la procédure de *leave-one-out cross-validation* (voir section 2.4.3) qui est employée pour évaluer la capacité de généralisation d'un masque d'entrée $\mathcal{M}$.

Cette méthode SFS commence par considérer chaque variable individuellement et sélectionne celle qui présente la meilleure capacité de généralisation. A toutes les autres étapes de l'algorithme, une variable supplémentaire est ajoutée à l'ensemble des variables sélectionnées. Cette variable est choisie de telle sorte que, ajoutée à l'ensemble des variables déjà sélectionnées, elle donne les meilleures performances de généralisation (voir algorithme 1).

A la fin, on dispose d'une série de d masques ainsi que de leur erreur de généralisation respective et on sélectionne le masque ayant la plus petite erreur.

2.6 Les familles d'ensembles d'hypothèses

Dans les sections précédentes, nous avons vu que Λ est un ensemble pouvant contenir les caractéristiques de n'importe quelle hypothèse, aussi bien linéaire que non-linéaire. Ensuite, un algorithme d'identification paramétrique applique le principe ERM et cherche dans l'ensemble Λ les caractéristiques α_N de l'hypothèse $h(\cdot, \alpha_N)$ minimisant le risque empirique. Pour faciliter l'identification paramétrique, il est fréquent de répartir les hypothèses en différentes familles de fonctions paramétriques.

Il existe, dans la littérature scientifique, différentes familles d'hypothèses comme par exemple les perceptrons multicouches [30, 31], les modèles linéaires [64] (voir section B à la page 169), les *k nearest neighbours* (k plus proches voisins) [62], les *lazy-learnings* [4, 15, 19, 14], les *Radial-Basis Function Networks*(RBFN) [30, 21], les *Multivariate Adaptive Regression Splines* (MARS) [36], etc...

Répartir les hypothèses en familles d'hypothèses permet de développer des algorithmes d'identification paramétrique spécifiques à chaque famille. Par exemple, l'algorithme du *least-square* [64] permet de réaliser l'identification paramétrique sur les familles d'hypothèses linéaires. Un autre exemple est l'algorithme de rétro-propagation du gradient [31] utilisé pour réaliser l'identification paramétrique des perceptrons multicouches. Les différentes familles d'hypothèses peuvent être réparties en deux grandes classes [18] : les familles d'hypothèses de type global et les familles d'hypothèses de type diviser pour conquérir.

2.6.1 Les familles d'hypothèses de type global

Dans cette famille d'hypothèses, la relation entre les entrées/sorties est décrite par une seule fonction dans tout le domaine des entrées $\mathcal{X}$. Les perceptrons multicouches [30, 31] et les hypothèses polynomiales [64] (dont les hypothèses linéaires font partie) sont deux exemples de familles d'hypothèses faisant partie de cette catégorie.

Pour certaines familles de type global, comme par exemple les perceptrons multicouches, l'identification paramétrique est un problème difficile d'optimisation non-linéaire. Nous avons constaté qu'il faut faire $\mathcal{K}$ identifications paramétriques et N prédictions pour réaliser une estimation de l'erreur de généralisation via les méthodes de *cross-validation* (voir section 2.4.3). Une augmentation du nombre d'identifications paramétriques $\mathcal{K}$ peut améliorer l'exactitude de l'estimateur de l'erreur de généralisation

G_N mais, comme une seule identification paramétrique peut être lente, la valeur de $\mathcal{K}$ ne peut généralement pas – dans ce cas – être élevée.

2.6.2 Les familles d'hypothèses de type diviser pour conquérir

Pour rendre la tâche de modélisation plus simple, nous divisons ici le domaine d'entrée $\mathcal{X}$ sur lequel nous cherchons à modéliser une fonction en un ensemble de sous-domaines d'entrée. Les familles de type diviser pour conquérir peuvent encore être classées en deux sous-familles : les familles d'hypothèses de type modulaire et de type local.

2.6.2.1 Les familles d'hypothèses de type modulaire

Dans ces familles, le domaine de modélisation $\mathcal{X}$ est divisé une seule fois lors de l'identification paramétrique. Chacun des sous-domaines devient ensuite un problème de modélisation classique. Sur les sous-domaines, il est possible d'utiliser des hypothèses plus simples mais la recherche des sous-domaines optimaux constitue quant à elle généralement un problème d'optimisation difficile. Les *Radial-Basis Function Networks*(RBFN) [30, 21] et les *Multivariate Adaptive Regression Splines*(MARS) [36] sont deux exemples de familles d'hypothèses faisant partie de cette catégorie.

2.6.2.2 Les familles d'hypothèses de type local

Lors de l'apprentissage sur des hypothèses locales, il suffit généralement de simplement mémoriser l'ensemble d'apprentissage. L'identification paramétrique est donc dans ce cas-ci très rapide. L'apprentissage est postposé et c'est uniquement lorsqu'une prévision est demandée en un point, appelé le *query-point*, que l'algorithme construit une fonction en utilisant les points les plus proches de ce *query-point*.

Les *k nearest neighbours*(KNN) [62] et les *lazy-learnings* [4, 15, 19, 14] sont deux exemples de familles d'hypothèses faisant partie de cette catégorie.

Les KNN sont l'une des familles d'hypothèses les plus simples (voir figure 2.13). La procédure des KNN est essentiellement composée des étapes suivantes :

- pour chaque *query-point* q, un ensemble de voisins est sélectionné à partir d'un critère, généralement la distance euclidienne [4],
- afin de construire l'hypothèse h, un ensemble d'hypothèses très simple est ensuite utilisé sur les voisins sélectionnés ; l'ensemble généralement utilisé est soit l'ensemble des hypothèses constantes soit l'ensemble des hypothèses linéaires,

4. Il est également fréquent d'utiliser comme critère la distance de Mahalanobis, basée sur la corrélation entre des variables d'entrée [58].

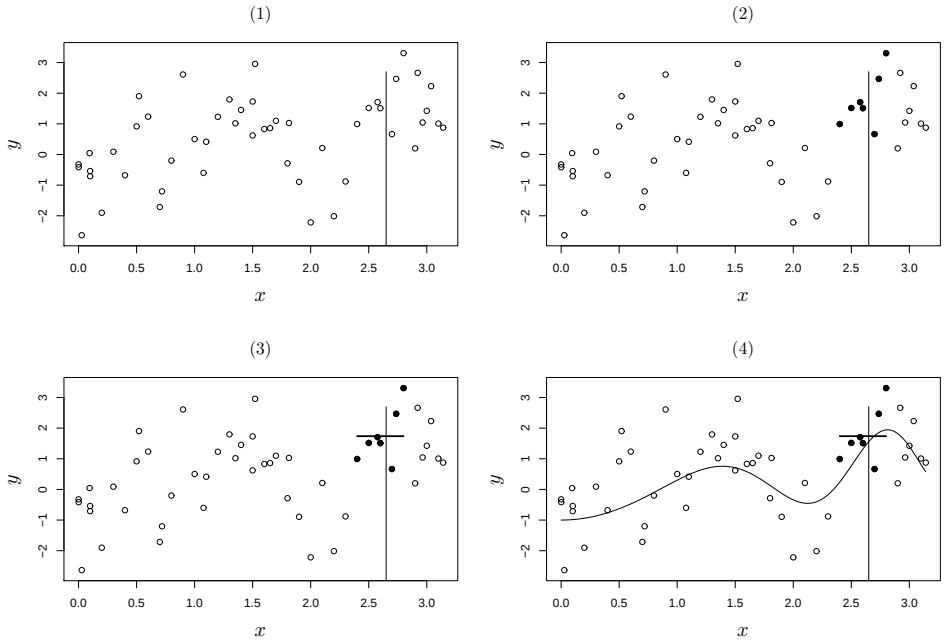


FIGURE 2.13 – (1) Nous utilisons les mêmes exemples d'apprentissage que ceux déjà utilisés à la page 20 et une prédiction est demandée à un KNN en 2.65. (2) Les sept points les plus proches sont sélectionnés et (3) c'est la moyenne de ces sept points qui est renvoyée comme prédiction. (4) La prédiction de l'hypothèse locale est 1.72 et la valeur de la fonction cible est 1.56. La prédiction est donc relativement proche de la vraie valeur.

– la prédiction de l'hypothèse de type KNN est alors la valeur de l'hypothèse en q.

Dans cette approche, le nombre de voisins utilisés pour construire le modèle local est fixé par l'utilisateur de l'algorithme. Il s'agit d'un paramètre structurel influant fortement sur les capacités de prédiction du modèle. Cette méthode du KNN est intrinsèquement adaptative car cette approche demande de garder constamment l'entièreté de l'ensemble d'apprentissage en mémoire. Dès lors, lorsqu'un nouvel exemple d'apprentissage est disponible, il suffit simplement de l'ajouter en mémoire pour mettre le modèle à jour.

Les *lazy-learnings* sont une instance particulière des familles d'hypothèses de type local où le nombre de voisins utilisés est choisi de manière automatique. L'idée consiste à commencer avec un petit nombre de voisins et ensuite la méthode ajoute de manière récursive de nouveaux voisins et ceci jusqu'au moment où la performance de prédiction du modèle local décroît. Cette approche permet la détection automatique de régions localement linéaires autour du *query-point*.

Pour estimer l'erreur de généralisation avec la technique de *cross-valida-*

tion (voir section 2.4.3), il est utile de faire $\mathcal{K}$ apprentissages et N prédictions. Comme dans le cas des modèles locaux, l'apprentissage est extrêmement rapide (il suffit de copier D_N en mémoire), diminuer la valeur de $\mathcal{K}$ n'est donc plus une solution d'accélération de la procédure de sélection de modèles. Dans le cas des hypothèses de type local, tout l'effort de calcul est reporté au moment de la demande d'une prédiction. Il faut donc plutôt diminuer le nombre de prédictions.

Cette thèse apportera des contributions dans le domaine des méthodes accélérant la procédure d'identification structurelle lorsque les hypothèses sont de type local. Soit $\Lambda_1, \ldots, \Lambda_k, \ldots, \Lambda_K$, une série de K classes d'hypothèses de type local et soit $e^i_{loo}(k)$, l'erreur de *leave-one-out* de la classe Λ_k sur l'exemple i de D_N. L'identification structurelle est un problème de type *selecting the best*. Elle consiste à trouver l'indice de la classe d'hypothèses minimisant l'erreur de généralisation. Afin de réaliser cette identification structurelle, il faut normalement faire $N \times K$ prédictions pour calculer les $N \times K$ erreurs de *leave-one-out*. Dans ce travail, nous étudierons des méthodes devant trouver l'*optimal* avec un budget d'erreurs de *leave-one-out* limité. Il faudra définir des stratégies choisissant, sur base des résultats précédents, la prochaine alternative à tester. Ce choix sera opéré afin d'obtenir une probabilité maximale de correctement sélectionner la classe d'hypothèses optimale en fin de procédure.

2.7 Conclusions

Ce chapitre a introduit les concepts de base de l'apprentissage supervisé qui seront utiles dans les prochains chapitres. La section 2.1 formalise le problème d'apprentissage supervisé. Elle définit les concepts : d'ensembles d'apprentissage D_N, d'hypothèses $h(\cdot, \alpha)$, des classes d'hypothèses Λ, de l'erreur fonctionnelle $R(\cdot)$ et de l'erreur empirique $R_{emp}(\cdot)$.

En apprentissage supervisé, le principe ERM est fondamental et préconise de choisir, pour une classe Λ, l'hypothèse $h(\cdot, \alpha_N)$ minimisant le risque empirique. La section 2.2 a montré qu'appliquer le principe ERM ne fonctionne pas dans tous les cas. Utiliser ce principe sur une classe Λ trop ou pas assez riche donne en effet de mauvais résultats.

L'erreur commise par un modèle choisi via le principe ERM peut être décomposée en une erreur d'estimation et une erreur d'approximation. Ces deux erreurs dépendent de la richesse de Λ et généralement lorsqu'une des erreurs diminue, l'autre augmente. Il faut donc trouver un juste compromis entre ces erreurs d'estimation et d'approximation. Cette tâche est réalisée par l'algorithme d'identification structurelle. Deux approches pour réaliser cette identification structurelle (la minimisation du risque structurel et la minimisation de l'erreur de généralisation) ont été présentées dans la section 2.3.

Après avoir défini formellement l'erreur de généralisation et après avoir présenté le compromis biais-variance, la section 2.4.1 a présenté différentes techniques pour estimer l'erreur de généralisation sur base d'une seule réalisation de $\mathbf{D}_N$. Ce sont les techniques par *cross-validation* qui seront utilisées dans ce travail.

La dernière section de ce chapitre a présenté quelques familles d'hypothèses. Dans le cas d'hypothèses de type local, diminuer le nombre d'apprentissages pour estimer l'erreur de généralisation n'a pas un grand impact sur la rapidité de l'algorithme de sélection de modèles. Il est préférable de diminuer le nombre de prédictions nécessaires. Cette thèse proposera, comme contributions, des techniques pour accélérer l'identification structurelle sur des familles d'hypothèses de type local.

Chapitre 3

Problèmes de sélection en environnement aléatoire

Ce chapitre porte sur les problèmes d'optimisation stochastique où l'objectif est de sélectionner – sur base d'un nombre limité d'observations – la variable aléatoire ayant l'espérance maximale.

Deux problèmes sont étudiés dans ce chapitre. Dans le premier problème, étant donné un nombre fini de tests, il s'agit de définir une séquence de tests qui, après avoir testé les alternatives, devra trouver – sur base des résultats collectés – l'alternative optimale (*selecting the best* [52, 25, 48, 65]). Le deuxième problème étudié est le *multi-armed bandit problem* [11]. Il s'agit également de définir une stratégie mais qui, cette fois, devra maximiser la somme des observations collectées durant les tests.

Une caractéristique commune de ces deux problèmes est de devoir déterminer l'alternative optimale le plus rapidement possible.

Dans le premier cas (*selecting the best*), l'algorithme doit optimiser son exploration afin de trouver, après avoir testé les différentes alternatives, l'alternative optimale en minimisant les tests nécessaires. Il s'agit d'un problème de pure exploration. L'algorithme peut donc tester des alternatives sous-optimales sans pénaliser le processus de sélection finale.

Le *multi-armed bandit problem* (MABP) se présente sous une forme différente. Ici aussi, il s'agit de trouver le plus rapidement possible l'alternative optimale mais, vu que la somme des observations collectées doit être maximisée, la manière d'obtenir cette alternative optimale est importante. Il ne s'agit plus d'un problème de pure exploration mais d'un juste compromis à trouver entre des actions orientées vers l'exploration et des actions orientées vers l'exploitation [60]. Les actions orientées vers l'exploration auront comme but de tester les alternatives non-familières afin d'améliorer les gains futurs, tandis que les actions orientées vers l'exploitation privilégieront plutôt les tests sur les alternatives qui avaient jusqu'alors donné les meilleurs résultats.

Ce chapitre étudie les deux types de problèmes et leurs relations. La section 3.1 introduit les notations utilisées et définit formellement le problème de sélection séquentielle en environnement aléatoire. La section 3.2 étudie le premier problème c'est-à-dire la sélection de l'optimum en un minimum de tests (*selecting the best*). La section 3.3 traite du second problème de sélection c'est-à-dire le *multi-armed bandit problem* (MABP).

3.1 Formalisation des problèmes de sélection séquentielle dans un environnement aléatoire

Considérons un problème d'optimisation stochastique composé de K variables aléatoires $\{\mathbf{z}_k\}_{k=1,\dots,K}$ de moyenne μ_k et d'écart-type σ_k. Nous dirons également que les variables aléatoires sont associées à K alternatives. Sauf mention contraire, nous supposerons que les variables $\mathbf{z}_k$ possèdent une distribution normale [83]. La variable $\mathbf{z}_k^i$ est définie comme la i-ième observation de la k-ième alternative et $\mathbf{Z}_k(n_k) = \left[\mathbf{z}_k^1, \mathbf{z}_k^2, \dots, \mathbf{z}_k^{n_k}\right]$ est un vecteur contenant n_k variables aléatoires indépendantes et identiquement distribuées.

Soit

$$[K] = \arg \max_{k \in [1,K]} \mu_k,$$

l'indice de la variable avec la plus grande moyenne et, de manière générale, soit $\mu_{[1]} < \mu_{[2]} < \ldots < \mu_{[K]}$ où $[k]$ représente l'indice de la variable ayant la k-ième plus petite moyenne. Dans cette thèse, nous supposons que les K moyennes sont toutes différentes.

Les deux types de problèmes étudiés ici devront chacun définir des stratégies séquentielles choisissant, sur base des résultats déjà collectés, la prochaine variable à tester [6].

L'objectif d'une stratégie qui résout le problème de sélection de l'optimum (*selecting the best*) est de choisir les alternatives qui seront examinées de telle sorte, qu'après cette phase de test, la variable sélectionnée sur base des observations collectées soit – si possible – l'optimum $[K]$ et ceci en un nombre de tests limité. Or, la nature aléatoire du problème ne donne pas la certitude de sélectionner l'optimum.

Ce sont donc plutôt des stratégies sélectionnant l'optimum avec une certaine probabilité $\Pr\{S_{[K]}\}$ qui seront définies. La quantité $\Pr\{S_{[K]}\}$ est la probabilité qu'une procédure de sélection de l'optimum, basée sur les résultats de la phase de test, choisisse l'alternative optimale $\mathbf{z}_{[K]}$ (en anglais *Probability of Correct Selection* (PCS)). Nous verrons dans la section 3.2 qu'après la phase de test, la sélection de l'optimum est basée sur un algorithme de type glouton [1]. Un algorithme glouton utilise les informa-

1. Dans la section 3.2.1, les algorithmes de Gupta [43] et de Nelson et al. [65] réalisent une maximisation des moyennes arithmétiques mais qui est chaque fois pénalisée de telle sorte que ces deux algorithmes sélectionnent généralement plusieurs variables.

tions collectées $\{\mathbf{Z}_1(n_1), \mathbf{Z}_2(n_2), \ldots, \mathbf{Z}_K(n_K)\}$ pour calculer les moyennes arithmétiques $\{\widehat{\boldsymbol{\mu}}_1, \widehat{\boldsymbol{\mu}}_2, \ldots, \widehat{\boldsymbol{\mu}}_K\}$ et sélectionne la variable $\mathbf{z}_{[\widehat{\mathbf{K}}]}$ présentant la plus grande moyenne arithmétique c'est-à-dire $[\widehat{\mathbf{K}}] = \arg\max_k \{\widehat{\boldsymbol{\mu}}_k\}$. Dans ce cas, la probabilité de faire une sélection correcte $\Pr\{S_{[K]}\}$ est la probabilité que la variable $\mathbf{z}_{[\widehat{\mathbf{K}}]}$ soit la variable optimale, c'est-à-dire

$$\Pr\{S_{[K]}\} = \Pr\left\{[\widehat{\mathbf{K}}] = [K]\right\}.$$

L'objectif d'une stratégie qui résout le *multi-armed bandit problem* est, quant à lui, de choisir les alternatives qui seront testées de telle sorte que la somme des observations collectées sur les alternatives soit maximale. A chaque étape, l'action entreprise par la stratégie doit être un juste compromis entre l'exploitation et l'exploration. Une action orientée exploitation aura comme but de maximiser le gain immédiat, une action de type glouton appartient donc aux stratégies dites de pure exploitation. Une action orientée exploration aura comme but de maximiser le gain futur obtenu via les prochaines actions d'exploitation. Pour maximiser les gains futurs, une solution consiste à utiliser la notion de $\Pr\{S_{[K]}\}$ car elle correspond à la probabilité qu'une action d'exploitation soit optimale. Les actions orientées exploration ont donc comme tâche d'augmenter cette probabilité.

3.2 Les stratégies d'échantillonnage pour la sélection de l'optimum

Cette section se penche sur la littérature existante concernant les algorithmes qui résolvent le premier type de problème, c'est-à-dire les problèmes de la sélection optimale avec un budget de tests.

Ces problèmes de sélection de l'optimum sont caractérisés par une première phase de test durant laquelle une stratégie est utilisée pour sélectionner séquentiellement les alternatives qui seront testées. C'est seulement après cette phase que l'information collectée sera utilisée pour sélectionner l'alternative optimale. Nous verrons que cette sélection de l'alternative optimale se fait via un algorithme de type glouton.

Différentes communautés scientifiques ont étudié ce problème de sélection de l'optimum. Cette section commence par présenter les approches provenant du domaine de la simulation Monte Carlo pour ensuite présenter celles issues du domaine de l'apprentissage artificiel. Les liens entre les approches de ces deux communautés seront mis en évidence à la fin de la section.

3.2.1 La sélection de sous-ensembles d'alternatives

La première approche (appelée en anglais *Single-Stage Screening Procedures* [65, 10]) teste durant une première phase N_0 fois chaque alternative

et construit ensuite un sous-ensemble $\mathbf{I}$ de $\{1, 2, \ldots, K\}$ qui contient $[K]$ avec une probabilité supérieure à $1 - \alpha_0$; c'est-à-dire $\Pr\left\{S^I_{[K]}\right\} \geq 1 - \alpha_0$. Idéalement, le nombre d'éléments dans l'ensemble $\mathbf{I}$ doit être petit et le meilleur cas de figure se présente lorsque l'ensemble $\mathbf{I}$ ne contient qu'un élément. Il est noté que, vu la nature aléatoire du problème de sélection, $\mathbf{I}$ est en général également aléatoire.

Dans cette première approche, les valeurs de N_0 et α_0 sont fixées par l'utilisateur. Comme N_0 est fixé au début de la procédure, il n'y a ici qu'une seule étape d'échantillonnage durant laquelle l'ensemble des $N_0 \times K$ observations est collecté. Durant la phase de test, il n'y a donc aucune possibilité pour l'algorithme de s'adapter, de manière automatique, au problème de sélection [2].

Il existe plusieurs manières de sélectionner des sous-ensembles de variables après la phase de test. Par exemple, la solution de Gupta [43] consiste à inclure dans $\mathbf{I}$ toutes les alternatives k tel que

$$\widehat{\boldsymbol{\mu}}_k \geq \max_{j \neq k} \widehat{\boldsymbol{\mu}}_j - \mathcal{H}_G \sigma \sqrt{\frac{2}{N_0}} \tag{3.1}$$

où $\widehat{\boldsymbol{\mu}}_j = 1/N_0 \cdot \sum_{i=1}^{N_0} \mathbf{z}_j^i$ est la moyenne arithmétique de N_0 observations générées à partir de $\mathbf{z}_j$ et où $\mathcal{H}_G$ est une valeur qui dépend du nombre K de variables et de la valeur α_0 [3]. Cette solution de Gupta suppose que tous les écarts-types sont connus et égaux $\sigma = \sigma_1 = \sigma_2 = \ldots = \sigma_K$. Il existe, dans [43], une variante de l'équation (3.1) qui suppose que l'écart-type est inconnu (mais tous les écarts-types sont toujours égaux entre eux) et dans laquelle

- σ est remplacé par son estimation $\widehat{\boldsymbol{\sigma}}_j$,
- $\mathcal{H}_G$ est ajusté pour tenir compte de l'incertitude de l'écart-type.

L'hypothèse d'égalité entre tous les écarts-types n'est, en pratique, généralement pas vérifiée. Dans [65], Nelson et al. proposent une nouvelle approche où les écarts-types sont inconnus et ne doivent pas être égaux entre eux. Il s'agit, bien entendu, de la situation la plus courante. Pour permettre à l'algorithme proposé par Nelson et al. de construire un ensemble $\mathbf{I}$, une hypothèse est posée quant à la différence minimum entre les deux meilleures alternatives. Nous supposons que la différence entre la meilleure moyenne $\mu_{[K]}$ et la seconde meilleure moyenne $\mu_{[K-1]}$ est au minimum de δ, c'est-à-dire que $\mu_{[K]} - \mu_{[K-1]} \geq \delta$. L'algorithme permet de construire un ensemble $\mathbf{I}$ contenant l'indice de l'alternative optimale $[K]$ avec une probabilité supérieure à $1 - \alpha_0$; c'est-à-dire $\Pr\left\{S^I_{[K]} \mid \mu_{[K]} - \mu_{[K-1]} \geq \delta\right\} \geq 1 - \alpha_0$.

L'utilisateur de la méthode proposée par Nelson et al. (voir l'algorithme 2) doit commencer par fixer les valeurs α_0, δ et N_0 (ligne 1). La valeur d'une

2. Durant la phase de test, il pourrait par exemple être utile de collecter plus d'informations sur les alternatives ayant une plus grande variance.

3. Voir [43] ou page 254 de [10] pour les tables de $\mathcal{H}_G$

Algorithme 2 Algorithme de *Nelson et al.* [65]

1: Fixer un niveau de confiance $1-\alpha_0$ avec $1/K < 1-\alpha_0 < 1$, la valeur de la zone d'indifférence δ et le nombre N_0 de tests qui vont être réalisés sur chaque variable aléatoire durant la phase d'échantillonnage.

2: Calculer $t = t_{(1-\alpha_0)^{\frac{1}{K-1}},\, N_0-1}$ où $t_{\beta,\nu}$ est le β-ième quantile d'une distribution t de *Student* avec ν degrés de liberté.

3: Échantillonner chaque variable aléatoire N_0 fois.

4: Calculer les estimations $\widehat{\boldsymbol{\mu}}$ et $\widehat{\boldsymbol{\sigma}}$ où

$$\widehat{\boldsymbol{\mu}}_k = \frac{1}{N_0}\sum_{i=1}^{N_0} \mathbf{z}_k^i \quad \text{et} \quad \widehat{\boldsymbol{\sigma}}_k = \sqrt{\frac{1}{N_0-1}\sum_{i=1}^{N_0}\left(\mathbf{z}_k^i - \widehat{\boldsymbol{\mu}}_k\right)^2}.$$

5: Calculer, pour tout $i \neq j$,

$$\mathbf{W}_{ij} = t\left(\frac{\widehat{\boldsymbol{\sigma}}_i}{N_0} + \frac{\widehat{\boldsymbol{\sigma}}_j}{N_0}\right)^{1/2}.$$

6: Construire l'ensemble $\mathbf{I}$:

$$\mathbf{I} = \left\{ i \text{ tel que } 1 \leq i \leq K \text{ et } \forall j \neq i,\ \widehat{\boldsymbol{\mu}}_i \geq \widehat{\boldsymbol{\mu}}_j - (\mathbf{W}_{ij} - \delta)^+ \right\}$$

où $y^+ = \max\{0, y\}$.

quantité t est ensuite calculée par l'algorithme comme indiqué à la ligne 2. Vient ensuite la phase de test (ligne 3) qui consiste à échantillonner chaque variable aléatoire N_0 fois. C'est sur base de ces observations qu'à la ligne 4 les estimateurs $\widehat{\boldsymbol{\mu}}_k$ et $\widehat{\boldsymbol{\sigma}}_k$ sont calculés pour chaque variable $\mathbf{z}_k$. Pour chaque couple d'indices de variables aléatoires (i, j) avec $i \neq j$, $\mathbf{W}_{ij}$ est calculé comme indiqué à la ligne 5. Et enfin, la dernière étape (ligne 6) consiste à construire l'ensemble $\mathbf{I}$.

L'algorithme de *Nelson et al.* a comme inconvénient que l'ensemble $\mathbf{I}$ possède généralement un cardinal supérieur à un. Cet algorithme sera réutilisé dans l'algorithme $\mathcal{S}+\mathcal{R}$ de la section 3.2.3 pour accélérer la sélection de l'optimum avec des méthodes en deux étapes d'échantillonnage.

Pour construire l'ensemble $\mathbf{I}$, les algorithmes de Gupta et de Nelson et al. réalisent tous les deux une maximisation des estimations des moyennes (voir l'équation 3.1 et la ligne 6 de l'algorithme 2) qui est chaque fois pénalisée (par $\mathcal{H}_G\sigma\sqrt{\frac{2}{N_0}}$ dans l'algorithme de Gupta ou par $(\mathbf{W}_{ij}-\delta)^+$ dans l'algorithme de Nelson et al.). Sans ces pénalisations, l'ensemble $\mathbf{I}$ aurait toujours un cardinal de un et contiendrait l'indice de la variable ayant la plus grande moyenne arithmétique. Plus la pénalisation est grande, plus le cardinal de $\mathbf{I}$ sera grand. Nous verrons dans la suite de cette section que tous les algo-

rithmes de sélection de l'optimum maximisent la moyenne arithmétique [4] et ceci avec une garantie $\Pr\{S_{[K]}\}$ que la variable optimale est celle avec la plus grande moyenne arithmétique. Il est à noter que si $|\mathbf{I}| = 1$ alors $\Pr\left\{S^I_{[K]}\right\}$ est équivalent à $\Pr\{S_{[K]}\}$.

3.2.2 Les procédures de sélection en deux étapes d'échantillonnage

Les deux méthodes précédentes présentent le désavantage que le cardinal de l'ensemble $\mathbf{I}$ est généralement plus grand que un. Il existe des procédures en une étape d'échantillonnage qui, lorsque les écarts-types sont connus, permettent de ne sélectionner qu'une seule variable avec une garantie $\Pr\{S_{[K]}\}$ que cette variable est bien optimale. Contrairement à la section précédente, N_0 n'est ici plus fixé par l'utilisateur mais directement par l'algorithme.

Bechhofer, dans [9], propose une procédure en une étape d'échantillonnage qui sélectionne une variable lorsque les écarts-types sont connus et égaux entre eux. La section 2.6 de [10], propose une autre procédure en une étape dans le cas où les écarts-types sont connus mais pas forcément égaux entre eux. Et enfin, Hayter, dans [45], propose une procédure qui sélectionne une variable dans le cas où les écarts-types $\sigma_1, \ldots, \sigma_K$ sont inconnus, ne sont pas nécessairement égaux entre eux mais sont bornés par une valeur connue ; c'est-à-dire $\max\{\sigma_1, \ldots, \sigma_K\} < \sigma_U$ où σ_U est connu.

Il n'existe pas de procédure en une étape d'échantillonnage qui permette de sélectionner une seule variable avec une garantie $\Pr\{S_{[K]}\}$ dans le cas où les écarts-types sont complètement inconnus [53]. Il faut alors au minimum deux étapes d'échantillonnage.

L'objet de cette section est l'étude des algorithmes en deux étapes pour la sélection de la variable optimale avec une garantie

$$\Pr\{S_{[K]} \mid \mu_{[K]} - \mu_{[K-1]} \geq \delta\} \geq 1 - \alpha_1$$

où δ et α_1 sont fixés par l'utilisateur. Dudewicz et al. [32] et Rinott [70] proposent des procédures en deux étapes d'échantillonnage dans le cas où les écarts-types $\sigma_1, \ldots, \sigma_K$ sont complètement inconnus. Comme déjà mentionné, ce cas est la situation la plus courante en pratique.

Les procédures de Dudewicz et al. et de Rinott commencent par échantillonner chaque alternative un petit nombre (N_0) de fois pour estimer leurs écarts-types. Ensuite, sur base (i) du nombre d'alternatives K, (ii) d'un niveau de confiance $1 - \alpha_1$, (iii) des estimations des écarts-types, (iv) du nombre N_0 de tests réalisés durant la première phase et (v) d'une zone d'indifférence δ, le nombre de tests total $\mathbf{n}_k$ nécessaires pour chaque alternative est calculé. Il est à noter que le nombre de tests total $\mathbf{n}_k$ devient une variable aléatoire car $\mathbf{n}_k$ est fonction des estimations des écarts-types. Enfin

4. Ils réalisent une action de type glouton.

sur base des résultats des $\mathbf{n}_k$ tests, les procédures sélectionnent la meilleure alternative. $[\widehat{\mathbf{K}}]$ est défini comme l'indice de la variable sélectionnée par l'algorithme après une maximisation des moyennes arithmétiques. Les algorithmes de sélection en deux étapes d'échantillonnage doivent donc définir, pour chaque alternative k, le nombre de tests $\mathbf{n}_k$ nécessaires pour garantir, sous une certaine probabilité, que l'alternative optimale $[K]$ possède bien la plus grande moyenne arithmétique.

Dans cette section, c'est la procédure de Rinott, plus simple à appliquer, qui est présentée. L'utilisateur de l'algorithme de Rinott (voir algorithme 3) définit le niveau de confiance $1 - \alpha_1$, la valeur de la zone d'indifférence δ et le nombre N_0 de tests qui vont être réalisés durant la première phase d'échantillonnage (ligne 1). Une quantité $\mathcal{H}_R$ [5] est ensuite fixée en fonction du nombre d'alternatives K, de α_1 et de N_0. Sur base de l'information collectée durant la première phase d'échantillonnage (ligne 3), les K estimations des écarts-types sont calculées à l'étape 4 pour être ensuite utilisées à l'étape 5 dans le but de déterminer le nombre total $\mathbf{n}_k$ de tests qui vont être réalisés sur chaque variable aléatoire. Vient ensuite la deuxième phase d'échantillonnage où chaque variable aléatoire est échantillonnée $\mathbf{n}_k - N_0$ fois (ligne 6). Sur base de toutes ces observations, les K estimations des moyennes sont calculées à l'étape 7 et, à la dernière étape, un algorithme glouton est appliqué.

En supposant que la différence entre la meilleure moyenne $\mu_{[K]}$ et la deuxième meilleure moyenne $\mu_{[K-1]}$ est d'au moins δ et que les variables aléatoires $\mathbf{z}_k$ sont distribuées selon une *normale*, l'algorithme de Rinott garantit, avec une confiance supérieure à $1 - \alpha_1$, que $[\widehat{\mathbf{K}}]$ est bien l'indice de la variable optimale.

Si le nombre d'alternatives K est trop grand, les procédures de sélection en deux étapes d'échantillonnage peuvent être inefficaces [65]. L'étape 5 de l'algorithme de Rinott (voir algorithme 3) définit $\mathbf{n}_k$, le nombre total de tests nécessaires pour chaque alternative, comme suit

$$\mathbf{n}_k = \max\left\{N_0, \left\lceil\left(\frac{\mathcal{H}_R \cdot \widehat{\boldsymbol{\sigma}}_k}{\delta}\right)^2\right\rceil\right\} \tag{3.4}$$

où N_0 est le nombre de tests initial, $\mathcal{H}_R$ est une constante dépendant de K, de α_1 et de N_0, $\widehat{\boldsymbol{\sigma}}_k$ est l'estimation de l'écart-type et δ est la zone d'indifférence.

L'équation (3.4) est basée sur l'hypothèse que l'on se trouve dans la pire des configurations c'est-à-dire celle où la meilleure moyenne $\mu_{[K]}$ est exactement à une distance δ de toutes les autres moyennes [6] (voir figure 3.1). C'est dans cette situation qu'il sera le plus difficile de trouver la variable optimale

5. Voir [88] ou pages 62-63 de [10] pour les tables de $\mathcal{H}_R$.

6. Le fait de supposer que l'on est dans le pire cas permet à l'équation (3.4) d'être indépendante des vraies moyennes [53].

Algorithme 3 Algorithme de *Rinott* [70]

1: Fixer un niveau de confiance $1-\alpha_1$ avec $1/K < 1-\alpha_1 < 1$, la valeur de la zone d'indifférence δ et le nombre N_0 de tests qui vont être réalisés sur chaque variable aléatoire durant la phase d'échantillonnage.
2: En fonction de K, de α_1 et de N_0 ; définir la valeur de $\mathcal{H}_R$.
3: Échantillonner chaque variable aléatoire N_0 fois.
4: Calculer les K estimations des écarts-types après la première phase d'échantillonnage

$$\widehat{\boldsymbol{\sigma}}_k = \sqrt{\frac{1}{N_0-1}\sum_{i=1}^{N_0}\left(\mathbf{z}_k^i - \frac{1}{N_0}\sum_{i=1}^{N_0}\mathbf{z}_k^i\right)^2}.$$

5: Déterminer le nombre total $\mathbf{n}_k$ de tests qui vont être réalisés sur chaque variable aléatoire

$$\mathbf{n}_k = \max\left\{N_0, \left\lceil\left(\frac{\mathcal{H}_R\cdot\widehat{\boldsymbol{\sigma}}_k}{\delta}\right)^2\right\rceil\right\}.$$

6: Échantillonner chaque variable aléatoire $\mathbf{n}_k - N_0$ fois.
7: Calculer les K estimations de la moyenne

$$\widehat{\boldsymbol{\mu}}_k = \frac{1}{\mathbf{n}_k}\sum_{i=1}^{\mathbf{n}_k}\mathbf{z}_k^i. \tag{3.2}$$

8: Sélectionner la variable qui possède la plus grande moyenne arithmétique

$$[\widehat{\mathbf{K}}] = \arg\max_k \widehat{\boldsymbol{\mu}}_k. \tag{3.3}$$

$\mathbf{z}_{[K]}$. Se placer dans le pire cas force l'algorithme à demander plus d'observations que nécessaire. Ce problème s'amplifie lorsque le nombre d'alternatives augmente et que leurs moyennes sont fort différentes. Il existe deux approches pour solutionner ce problème.

La première approche consiste à combiner les algorithmes de *screening* en une étape (voir section 3.2.1) avec les algorithmes de sélection en deux étapes (comme la méthode de Rinott) dans le but de supprimer, avant la deuxième phase d'échantillonnage, les alternatives les moins compétitives. Cette approche est présentée à la section 3.2.3.

La deuxième approche est une extension naturelle de la première. Plutôt que d'avoir deux étapes d'échantillonnage, cette deuxième approche propose une méthode complètement séquentielle. De telles méthodes séquentielles échantillonnent les alternatives encore en jeu, suppriment sur base des ob-

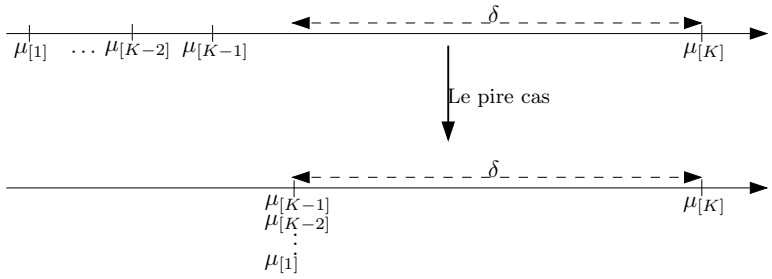


FIGURE 3.1 – La distance entre la meilleure moyenne $\mu_{[K]}$ et la seconde meilleure moyenne $\mu_{[K-1]}$ est au minimum de δ. Pour calculer $\mathbf{n}_k$ (voir équation (3.4)) l'algorithme de Rinott suppose être dans le pire cas. Connaissant δ, le pire cas correspond à la situation où la distance entre la meilleure moyenne $\mu_{[K]}$ et toutes les autres moyennes est exactement de δ.

servations collectées les alternatives les moins compétitives et continuent de la sorte. Cette approche est présentée à la section 3.2.4.

3.2.3 Les procédures combinées pour l'optimisation de la sélection en deux étapes d'échantillonnage

Pour ne pas augmenter inutilement le nombre de tests nécessaires, un algorithme en deux étapes (tel l'algorithme de Rinott, présenté dans la section 3.2.2) peut être appliqué sur les alternatives restant en course après la première phase d'une procédure de *screening* (voir section 3.2.1) [65].

Soit $\mathcal{S}$ une procédure de *screening* (voir section 3.2.1) qui teste N_0 fois les K alternatives et qui sur base de ces observations, construit un sous-ensemble $\mathbf{I}$ de $\{1, \ldots, K\}$ contenant $[K]$ avec $\Pr\{S_{[K]}\} \geq 1 - \alpha_0$. Soit $\mathcal{R}$ une procédure de sélection en deux étapes d'échantillonnage (voir section 3.2.2) qui sélectionne l'alternative optimale $[K]$ avec une probabilité $\Pr\{S_{[K]}\} \geq 1 - \alpha_1$ en supposant que $\mu_{[K]} - \mu_{[K-1]} \geq \delta$. Dans cette section, la procédure de *screening* $\mathcal{S}$ est combinée avec une procédure $\mathcal{R}$ de sélection en deux étapes. La procédure $\mathcal{R}$ utilise l'ensemble $\mathbf{I}$ créé par $\mathcal{S}$ (et les observations utilisées pour sélectionner $\mathbf{I}$) pour initialiser la première étape de $\mathcal{R}$. Si $\mu_{[K]} - \mu_{[K-1]} \geq \delta$, alors la probabilité de faire une sélection correcte avec une procédure combinée $\mathcal{S} + \mathcal{R}$ est plus grande que $1 - (\alpha_0 + \alpha_1)$ [65] c'est-à-dire

$$\Pr\{S_{[K]}\} \geq 1 - (\alpha_0 + \alpha_1) = 1 - \alpha_g.$$

Généralement α_0 et α_1 sont fixés à la même valeur et donc $\alpha_0 = \alpha_1 = \alpha_g/2$.

L'utilisateur de l'algorithme $\mathcal{S} + \mathcal{R}$ (voir algorithme 4) doit commencer par définir les valeurs de α_0 , α_1, δ et N_0 (ligne 1). Ensuite les valeurs de t et $\mathcal{H}_R$ sont calculées (ligne 2). Après cette partie d'initialisation vient la première phase d'échantillonnage et, sur base de ces observations, les estimations des K moyennes $\widehat{\boldsymbol{\mu}}_k$ et des K écarts-types $\widehat{\boldsymbol{\sigma}}_k$ sont définies (ligne 3).

Algorithme 4 Algorithme $\mathcal{S}+\mathcal{R}$ [65]

1: Fixer un niveau de confiance global $1-\alpha_g$ avec $1/K < 1-\alpha_g < 1$, les valeurs de α_0 et α_1 avec $\alpha_0+\alpha_1=\alpha_g$, la valeur de la zone d'indifférence δ et la valeur de N_0.

2: Calculer $t = t_{(1-\alpha_0)^{\frac{1}{K-1}},\ N_0-1}$ où $t_{\beta,\nu}$ est le β-ième quantile d'une distribution t de *Student* avec ν degrés de liberté et définir la valeur de $\mathcal{H}_R$ en fonction de K, de α_1 et de N_0.

3: Échantillonner chaque variable aléatoire N_0 fois et calculer les estimations $\widehat{\boldsymbol{\mu}}_k$ et $\widehat{\boldsymbol{\sigma}}_k$.

4: Pour tout $i \neq j$, calculer $\mathbf{W}_{ij} = t\left(\frac{\widehat{\boldsymbol{\sigma}}_i}{N_0} + \frac{\widehat{\boldsymbol{\sigma}}_j}{N_0}\right)^{1/2}$ et ensuite construire l'ensemble $\mathbf{I}$ avec $y^+ = \max\{0, y\}$:

$$\mathbf{I} = \left\{ i \ \text{ tel que } \ 1 \leq i \leq K \ \text{ et } \ \forall j \neq i,\ \widehat{\boldsymbol{\mu}}_i \geq \widehat{\boldsymbol{\mu}}_j - (\mathbf{W}_{ij} - \delta)^+ \right\}.$$

5: Si $\mathbf{I}$ ne contient qu'un indice, renvoyer cet indice sinon déterminer le nombre total $\mathbf{n}_k$ de tests qui vont être réalisés pour tout k dans $\mathbf{I}$

$$\mathbf{n}_k = \max\left\{ N_0, \left\lceil \left(\frac{\mathcal{H}_R \cdot \widehat{\boldsymbol{\sigma}}_k}{\delta} \right)^2 \right\rceil \right\}.$$

6: Échantillonner $\mathbf{n}_k - N_0$ fois chaque variable aléatoire $\mathbf{z}_k$ avec $k \in \mathbf{I}$.

7: Calculer les estimations de la moyenne $\widehat{\boldsymbol{\mu}}_k$ avec $k \in \mathbf{I}$ et sélectionner dans $\mathbf{I}$ la variable avec la plus grande moyenne arithmétique $[\widehat{\mathbf{K}}] = \arg\max_{k \in \mathbf{I}} \widehat{\boldsymbol{\mu}}_k$.

Pour tous les indices i et j tel que $i \neq j$, on utilise la quantité t et les estimations des écarts-types pour calculer $\mathbf{W}_{ij}$ et construire l'ensemble $\mathbf{I}$ (ligne 4). Cet ensemble correspond à l'ensemble qui aurait été créé par l'algorithme de *screening* de Nelson et al. (voir l'algorithme 2). Si l'ensemble $\mathbf{I}$ contient l'indice d'une seule variable, l'algorithme s'arrête et sélectionne cet indice. Dans le cas contraire, il faut, pour chaque indice de l'ensemble $\mathbf{I}$, calculer le nombre total de tests $\mathbf{n}_k$ nécessaires (ligne 5). Vient ensuite la deuxième phase d'échantillonnage (ligne 6) où chaque variable présente dans $\mathbf{I}$ est testée $\mathbf{n}_k - N_0$ fois. Les observations des variables encore dans l'ensemble $\mathbf{I}$ sont finalement utilisées pour calculer leur moyenne arithmétique. La variable possédant la moyenne arithmétique la plus élevée est sélectionnée par l'algorithme.

L'algorithme $\mathcal{S}+\mathcal{R}$ sera comparé expérimentalement avec d'autres méthodes dans le chapitre 5.

3.2.4 Les procédures séquentielles pour l'optimisation de la sélection

Cette section présente les techniques séquentielles visant à la sélection de l'optimum en un minimum de tests. Ces techniques ont pour but de réduire l'effort de calcul nécessaire pour trouver l'optimum en éliminant aussi vite que possible les alternatives les moins performantes. Contrairement à l'algorithme $\mathcal{S}+\mathcal{R}$ présenté dans la section précédente, les algorithmes séquentiels rencontrent un nombre beaucoup plus important d'occasions d'éliminer les alternatives sous-optimales.

Ces techniques commencent par initialiser un ensemble $\mathbf{I}_1$ avec toutes les alternatives[7]. Ensuite, à chaque étape $l = 1, 2, 3, \ldots$, les alternatives qui sont encore dans l'ensemble $\mathbf{I}_l$ sont échantillonnées et les moins performantes sont retirées de l'ensemble pour construire l'ensemble $\mathbf{I}_{l+1}$. L'algorithme se poursuit tant que, par exemple, le cardinal de l'ensemble $\mathbf{I}_{l+1}$ est supérieur à un.

Comme déjà mentionné dans ce chapitre, le problème du choix optimal en un minimum de tests constitue une tâche classique. On le retrouve dans le domaine de la simulation où la configuration optimale doit être sélectionnée en un minimum de tests, ou encore dans le domaine de l'apprentissage artificiel où la classe d'hypothèses locales optimales doit être trouvée en réalisant le moins de prédictions possible. Chacune des deux communautés a proposé ses algorithmes de sélection de l'optimum. Cette section commence par présenter l'algorithme $\mathcal{KN}$ [51] qui est un algorithme séquentiel proposé pour résoudre les problèmes de sélection en simulation. Ensuite les algorithmes séquentiels Hoeffding-*race* [59] et F-race [16, 13], des classiques en apprentissage artificiel, seront présentés. La section met également en évidence les liens entre les deux approches.

3.2.4.1 La procédure $\mathcal{KN}$

La procédure $\mathcal{KN}$ [51] est une procédure purement séquentielle. Si la différence entre la meilleure moyenne $\mu_{[K]}$ et la seconde meilleure moyenne $\mu_{[K-1]}$ est d'au moins δ, cette procédure garantit, avec une confiance plus grande ou égale à $1 - \alpha_{\mathcal{KN}}$, que la variable sélectionnée a la plus grande moyenne ; c'est-à-dire $\Pr\{S_{[K]} \mid \mu_{[K]} - \mu_{[K-1]} \geq \delta\} \geq 1 - \alpha_{\mathcal{KN}}$.

L'algorithme 5 décrit les étapes de $\mathcal{KN}$. Après l'initialisation des valeurs de $\alpha_{\mathcal{KN}}$, de δ et de N_0 par l'utilisateur (ligne 1), l'algorithme $\mathcal{KN}$ construit l'ensemble initial $\mathbf{I}_1$ et calcule la valeur de la quantité h^2 (ligne 2). Vient ensuite la phase d'échantillonnage qui construit les K estimations des moyennes $\widehat{\boldsymbol{\mu}}_k$ (ligne 3). Ces K estimations sont utilisées pour calculer l'estimation de la variance de la différence entre les alternatives i et j pour tout $i \neq j$ (ligne 4). C'est l'étape 5 de l'algorithme qui élimine les alterna-

7. Vu la façon d'initialiser l'ensemble $\mathbf{I}_1$, $\Pr\{\mathbf{I}_1 = \{1, 2, \ldots, K\}\} = 1$.

Algorithme 5 Algorithme $\mathcal{KN}$ [51]

1: Fixer un niveau de confiance global $1-\alpha_{\mathcal{KN}}$ avec $1/K < 1-\alpha_{\mathcal{KN}} < 1$, la valeur de la zone d'indifférence δ et le nombre N_0.
2: Initialiser :

$$\mathbf{I}_1 = \{1, 2, \ldots, K\}$$

et

$$h^2 = \left[(2\alpha_{\mathcal{KN}}/(K-1))^{-2/(N_0-1)} - 1\right](N_0-1).$$

3: Échantillonner chaque variable aléatoire N_0 fois et poser :$n \leftarrow N_0$.
4: Calculer pour tous i et j tel que $i \neq j$

$$\mathbf{S}_{ij}^2 = \frac{1}{N_0-1}\sum_{m=1}^{N_0}\left(\mathbf{z}_i^m - \mathbf{z}_j^m - \left[\widehat{\boldsymbol{\mu}}_i - \widehat{\boldsymbol{\mu}}_j\right]\right)^2.$$

5: Construire l'ensemble $\mathbf{I}_{l+1}$

$$\mathbf{I}_{l+1} = \left\{i \ : \ i \in \mathbf{I}_l \ \text{ et } \ \widehat{\boldsymbol{\mu}}_i \geq \widehat{\boldsymbol{\mu}}_j - \mathbf{W}_{ij} \ , \ \ \forall j \in \mathbf{I}_l \ , \ j \neq i\right\}$$

avec $\mathbf{W}_{ij} = \max\left\{0, \frac{\delta}{2l}\left(\frac{h^2\mathbf{S}_{ij}^2}{\delta^2} - l\right)\right\}$.
6: **si** $|\mathbf{I}_{l+1}| = 1$ **alors**
7: Sélectionner la valeur restant dans l'ensemble $\mathbf{I}_{l+1}$.
8: **sinon**
9: $\forall k \in \mathbf{I}_{l+1}$, échantillonner les alternatives $\mathbf{z}_k$.
10: $l \leftarrow l+1$ et $n \leftarrow n+1$.
11: Retourner à l'étape 5.
12: **fin si**

tives les moins performantes et construit l'ensemble $\mathbf{I}_{l+1}$. S'il ne reste plus qu'une alternative dans l'ensemble $\mathbf{I}$ (ligne 6), cela signifie que la tâche de sélection de l'optimum est terminée et l'algorithme donne la valeur restant dans $\mathbf{I}$. Dans le cas contraire, chaque alternative demeurant en compétition est testée (ligne 9) et l'algorithme recommence la construction d'un nouvel ensemble $\mathbf{I}$ (ligne 11).

3.2.4.2 La procédure Hoeffding-*race*

Une autre famille de procédures séquentielles pour l'optimisation de la sélection sont les algorithmes de *racing* [16, 20, 59, 89] utilisés en apprentissage artificiel pour la sélection de la classe d'hypothèses locales optimales en un minimum de tests de *leave-one-out*.

Deux versions de l'algorithme de *racing* sont ici étudiées, le Hoeffding-*race* [59] et le *F-race* [16, 13]. Il est à noter, que ces deux algorithmes ne

supposent pas que les variables $\mathbf{z}_1, \ldots, \mathbf{z}_K$ suivent des distributions normales. Commençons par le Hoeffding-*race*.

A chaque étape du Hoeffding-*race*, les alternatives encore en course sont testées et les moyennes arithmétiques $\widehat{\boldsymbol{\mu}}_k$ sont mises à jour. Cet algorithme se base sur le principe que plus le nombre d'observations est grand, plus les moyennes arithmétiques $\widehat{\boldsymbol{\mu}}_k$ sont proches des vraies moyennes μ_k. Pour mesurer cet écart, l'algorithme Hoeffding-*race* utilise une borne statistique appelée borne de Hoeffding (voir annexe E).

La borne de Hoeffding est très générale et suppose uniquement que, pour une alternative k, les n_k observations $\mathbf{z}_k^1, \mathbf{z}_k^2, \ldots, \mathbf{z}_k^{n_k}$ sont indépendantes et identiquement distribuées [8]. Dans ce cas, la probabilité que la distance entre $\widehat{\boldsymbol{\mu}}_k$ et μ_k soit plus grande que ϵ est bornée supérieurement de telle sorte que

$$\Pr\{|\widehat{\boldsymbol{\mu}}_k - \mu_k| > \epsilon\} < 2e^{-2n_k\epsilon^2/B^2} \tag{3.5}$$

où B est une borne supérieure sur $\mathbf{z}$.

En fixant la partie droite de l'équation (3.5) comme égale à α_h et en mettant ϵ en évidence ; on obtient

$$\epsilon(n_k) = \sqrt{\frac{B^2 \ln(2/\alpha_h)}{2n_k}} \tag{3.6}$$

où n_k est le nombre de tests réalisés sur l'alternative k. Et comme l'équation (3.5) est équivalente à

$$\Pr\{|\widehat{\boldsymbol{\mu}}_k - \mu_k| \leq \epsilon\} \geq 1 - \alpha_h,$$

l'intervalle

$$[\widehat{\boldsymbol{\mu}}_k - \epsilon(n_k) \ , \ \widehat{\boldsymbol{\mu}}_k + \epsilon(n_k)] \tag{3.7}$$

contient donc μ_k avec une probabilité supérieure à $1 - \alpha_h$.

Dans un problème de maximisation, les alternatives dont la borne supérieure est plus petite que la borne inférieure maximum sont éliminées par l'algorithme Hoeffding-*race* pour ne plus jamais être testées (voir figure 3.2).

Les alternatives restantes sont testées en parallèle et l'algorithme utilise des bornes statistiques pour déterminer si une alternative est significativement inférieure aux autres. Si le cas se présente, cette alternative détectée comme significativement inférieure sera retirée de la « course » pour ne plus jamais être testée. Ceci permet de concentrer la puissance de calcul sur les alternatives les plus performantes et de pouvoir donc, plus rapidement, trouver l'optimum. Cet algorithme réitère la procédure, en sélectionnant à chaque fois de nouvelles observations, jusqu'à ce que toutes les alternatives

8. Dans les procédures séquentielles pour la sélection de l'optimum, n_k n'est plus aléatoire, on a $n_k = N_0 + l - 1$.

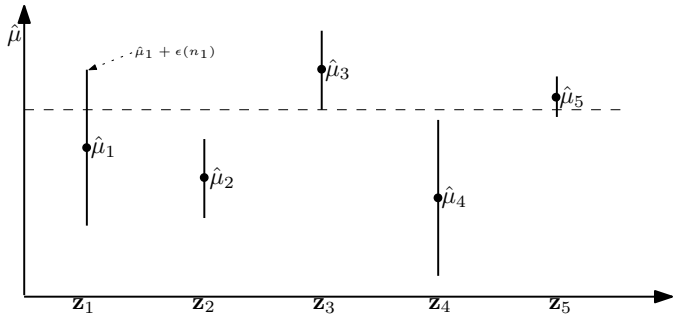


FIGURE 3.2 – Exemple où les intervalles de confiance des gains des cinq alternatives sont comparés. Ici, la borne inférieure de l'alternative trois élimine les alternatives deux et quatre car leurs bornes supérieures sont en deçà de la borne inférieure maximum.

aient été éliminées sauf une ou qu'un nombre d'observations suffisant (par exemple H_{max}) ait été sélectionné et dans ce cas, l'alternative avec la plus grande moyenne arithmétique est choisie.

Après avoir initialisé les variables α_h, B, H_{max} et N_0 (ligne 1), le Hoeffding-*race* (algorithme 6) construit un ensemble $\mathbf{I}$ contenant toutes les alternatives (ligne 2) et teste chaque alternative N_0 fois (ligne 3). Ensuite, à chaque étape de l'algorithme, les bornes des alternatives encore dans $\mathbf{I}$ sont construites (ligne 4) et utilisées pour supprimer les alternatives les moins performantes (ligne 5). L'algorithme Hoeffding-*race* s'arrête dans deux cas : s'il ne reste plus qu'une alternative dans l'ensemble $\mathbf{I}$, cette alternative est alors sélectionnée (lignes 6 et 7) ou si le nombre de tests réalisés est supérieur à une borne H_{max} fixée par l'utilisateur, l'alternative dans $\mathbf{I}$ possédant la plus grande espérance arithmétique est alors sélectionnée (lignes 8 et 9). Dans tous les autres cas, l'algorithme Hoeffding-*race* échantillonne les alternatives encore en course et poursuit la recherche en reprenant à l'étape 4.

La quantité $1-\alpha_h$ est, pour une seule étape et pour une seule alternative, une borne inférieure de la certitude que μ_k est dans $[\widehat{\boldsymbol{\mu}}_k - \epsilon(n_k)\ ,\ \widehat{\boldsymbol{\mu}}_k + \epsilon(n_k)]$. Il ne s'agit donc pas d'une garantie, pourtant souhaitée, que la procédure de *racing* trouve l'alternative optimale. Durant les l étapes, Maron et Moore à la section 3.1.4 de [59] montrent que la probabilité, que les K moyennes μ_k soient dans les intervalles correspondants $[\widehat{\boldsymbol{\mu}}_k - \epsilon(n_k)\ ,\ \widehat{\boldsymbol{\mu}}_k + \epsilon(n_k)]$, est plus grande ou égale à $1 - \alpha_h \cdot l \cdot K$; c'est-à-dire

$$\Pr\{R\} \geq 1 - \alpha_h \cdot l \cdot K$$

où R correspond à l'événement « *les K moyennes sont, durant les l étapes, dans leur intervalle* » .

Supposons que H_{max} est suffisamment grand de telle sorte que l'algorithme s'est arrêté parce que $|\mathbf{I}| = 1$. Etant donné que l'événement R est un

Algorithme 6 Algorithme Hoeffding-*race* [59]

1: Fixer le niveau de confiance des bornes statistiques $1-\alpha_h$, les valeurs B et H_{max} et le nombre $N_0 \geq 2$ de tests qui vont être réalisés sur chaque variable aléatoire durant la première phase d'échantillonnage.
2: Initialiser : $\mathbf{I}_1 = \{1, 2, \ldots, K\}$ et $l \leftarrow 1$.
3: Échantillonner chaque variable aléatoire N_0 fois.
4: Calculer les intervalles de confiance (3.7) via l'équation (3.6) des alternatives qui sont dans l'ensemble $\mathbf{I}_l$.
5: Pour construire l'ensemble $\mathbf{I}_{l+1}$, les alternatives de $\mathbf{I}_l$, dont la borne supérieure ne dépasse pas la borne inférieure maximum, sont supprimées.
6: **si** $|\mathbf{I}_{l+1}| = 1$ **alors**
7: Sélectionner la valeur dans l'ensemble $\mathbf{I}_{l+1}$.
8: **sinon si** le nombre d'observations collectées est $> H_{max}$ **alors**
9: Sélectionner l'alternative dans $\mathbf{I}_{l+1}$ ayant la plus grande espérance arithmétique.
10: **sinon**
11: $\forall k \in \mathbf{I}_{l+1}$, échantillonner les alternatives $\mathbf{z}_k$.
12: $l \leftarrow l + 1$.
13: Retourner à l'étape 4.
14: **fin si**

événement plus strict qui inclut l'événement $S_{[K]}$ (≪ *l'algorithme sélectionne l'optimal* ≫), on obtient

$$\Pr\{S_{[K]}\} \geq \Pr\{R\} \geq 1 - \alpha_h \cdot l \cdot K.$$

Il existe donc aussi une borne inférieure sur $\Pr\{S_{[K]}\}$ dans le cas du *racing* et il y a de ce fait une relation entre les algorithmes de sélection séquentielle proposés par les deux communautés (simulation et apprentissage artificiel). Dans le chapitre 5, nous proposons, comme contribution de cette thèse, un algorithme de sélection séquentielle qui, à chaque étape, teste les alternatives dans le but explicite de faire augmenter cette quantité $\Pr\{S_{[K]}\}$.

3.2.4.3 La procédure *F-race*

Une autre version de l'algorithme du *racing* utilisée dans cette thèse est basée sur le test de *Friedman* [26]. C'est dans [16] que le test de *Friedman* a été utilisé pour la première fois dans un algorithme de *racing*, dénommé *F-race*.

Pour décrire l'algorithme *F-race*, supposons que $\mathbf{I}_l$ contienne les index des alternatives non-éliminées au début de l'étape l et que $|\mathbf{I}_l|$ soit le nombre d'alternatives encore en course. L'étape l commence par échantillonner toutes les variables aléatoires dans $\mathbf{I}_l$ pour construire l'ensemble $\left\{\mathbf{z}_k^{l+1}\right\}$ avec $k \in \mathbf{I}_l$.

Cet ensemble $\left\{\mathbf{z}_1^{l+1}, \ldots, \mathbf{z}_k^{l+1}, \ldots, \mathbf{z}_{|\mathbf{I}_l|}^{l+1}\right\}$ avec $k \in \mathbf{I}_l$ est appelé bloc. Le test de *Friedman* est un test non-paramétrique basé sur le calcul des rangs dans différents blocs. Dans chaque bloc i, $\left\{\mathbf{z}_1^i, \ldots, \mathbf{z}_k^i, \ldots, \mathbf{z}_{|\mathbf{I}_l|}^i\right\}$ avec $k \in \mathbf{I}_l$, les quantités $\mathbf{z}_k^i$ sont classées de la plus grande à la plus petite où $\mathbf{r}_{ki}$ est le rang de l'alternative k dans le bloc i et $\mathbf{r}_k = \sum_{i=1}^{l+1} \mathbf{r}_{ki}$ est la somme des rangs de l'alternative k.

Le test de *Friedman* considère la statistique suivante :

$$\mathbf{T} = \frac{(|\mathbf{I}_l| - 1) \sum_{k \in \mathbf{I}_l} \left(\mathbf{r}_k - \frac{l(|\mathbf{I}_l|+1)}{2}\right)^2}{\sum_{i=1}^{l} \sum_{k \in \mathbf{I}_l} (\mathbf{r}_{ki})^2 - \frac{l|\mathbf{I}_l|(|\mathbf{I}_l|+1)^2}{4}}. \tag{3.8}$$

Sous l'*hypothèse nulle* que tous les classements possibles dans les blocs sont équiprobables, $\mathbf{T}$ a une distribution khi-carré avec $|\mathbf{I}_l| - 1$ degré de liberté. Si la statistique $\mathbf{T}$ dépasse le $1 - \alpha_f$ quantile d'une telle distribution, alors l'*hypothèse nulle* est rejetée avec un niveau α_f en faveur de l'hypothèse qu'au moins une alternative est supérieure aux autres.

Si l'*hypothèse nulle* est rejetée, une comparaison entre toutes les paires d'alternatives i et j dans $\mathbf{I}_l$ est exécutée. Les alternatives i et j sont considérées comme différentes si

$$\frac{|\mathbf{r}_i - \mathbf{r}_j|}{\sqrt{\frac{2l\left(1 - \frac{\mathbf{T}}{l(|\mathbf{I}_l|-1)}\right)\left(\sum_{i=1}^{l} \sum_{k \in \mathbf{I}_l} (\mathbf{r}_{ki})^2 - \frac{l|\mathbf{I}_l|(|\mathbf{I}_l|+1)^2}{4}\right)}{(l-1)(|\mathbf{I}_l|-1)}}} > t_{1-\alpha_f/2} \tag{3.9}$$

où $t_{1-\alpha_f/2}$ est le $1 - \alpha_f/2$ quantile d'une distribution de *Student*.

L'utilisateur doit commencer par initialiser les trois variables α_f, H_{max} et N_0 (ligne 1). Le F-*race* (algorithme 7) construit ensuite un ensemble $\mathbf{I}$ contenant toutes les alternatives (ligne 2). Après cette phase d'initialisation, chaque alternative est échantillonnée N_0 fois (ligne 3). C'est à l'étape 4 que l'algorithme teste s'il existe au moins une alternative significativement supérieure aux autres. Si tel est le cas, l'algorithme cherche les alternatives sous-optimales (ligne 6), tandis que dans le cas contraire, comme il n'y a pas d'alternative significativement différente, l'ensemble $\mathbf{I}$ n'est pas modifié (ligne 8). L'algorithme F-*race* s'arrête dans deux cas : lorsque l'ensemble $\mathbf{I}$ ne contient plus qu'une alternative (ligne 11) ou lorsque le nombre de tests effectués est supérieur à une borne H_{max} fixée par l'utilisateur (ligne 13). Dans tous les autres cas, l'algorithme F-*race* échantillonne les alternatives non éliminées et recommence la recherche à l'étape 4.

Dans le F-*race*, l'utilisation des rangs à un rôle important pour deux raisons [16] :

– La première raison est liée à la nature non-paramétrique des tests basés sur les rangs. Le principal avantage des analyses non-paramétriques réside dans le fait qu'il n'est pas utile de faire d'hypothèses quant à

Algorithme 7 Algorithme F-*race* [16]

1: Fixer le niveau de confiance des bornes statistiques $1 - \alpha_f$, la valeur de H_{max} et le nombre $N_0 \geq 2$ d'observations qui vont être collectées sur chaque variable aléatoire durant la première phase d'échantillonnage.
2: Initialiser : $\mathbf{I}_1 = \{1, 2, \ldots, K\}$ et $l \leftarrow 1$.
3: Échantillonner chaque variable aléatoire N_0 fois.
4: Tester, via l'équation (3.8), si $\mathbf{I}_l$ contient au moins une alternative significativement supérieure aux autres.
5: **si** il y a au moins une alternative significativement supérieure aux autres **alors**
6: Pour construire l'ensemble $\mathbf{I}_{l+1}$, l'équation (3.9) est appliquée sur toutes les paires d'alternatives de l'ensemble $\mathbf{I}_l$.
7: **sinon**
8: $\mathbf{I}_{l+1} \leftarrow \mathbf{I}_l$.
9: **fin si**
10: **si** $|\mathbf{I}_{l+1}| = 1$ **alors**
11: Sélectionner la valeur dans l'ensemble $\mathbf{I}_{l+1}$.
12: **sinon si** le nombre d'observations collectées est $> H_{max}$ **alors**
13: Sélectionner l'alternative contenue dans $\mathbf{I}_{l+1}$ présentant la plus grande espérance arithmétique.
14: **sinon**
15: $\forall k \in \mathbf{I}_{l+1}$, échantillonner les alternatives $\mathbf{z}_k$.
16: $l \leftarrow l + 1$.
17: Retourner à l'étape 4.
18: **fin si**

la distribution des observations. Néanmoins, lorsque l'hypothèse d'un test paramétrique se vérifie, celui se révèle généralement meilleur que la version non-paramétrique.

– La deuxième raison est liée au fait que l'utilisation des rangs implémente de manière naturelle les techniques d'analyses par blocs. Par exemple, considérons les problèmes de sélection de modèles par *leave-one-out cross-validation* (section 2.4.3) où $e^i_{loo}(\Lambda_k)$ désigne l'erreur de *leave-one-out* calculée sur $\langle x_i, y_i \rangle$ avec la classe d'hypothèses Λ_k [9]. La variabilité observée dans $e^i_{loo}(\Lambda_k)$ est due à différentes sources : l'erreur est intrinsèquement aléatoire, les instances $\langle x_i, y_i \rangle$ sur lesquelles les erreurs sont calculées peuvent être très différentes et certaines classes d'hypothèses sont meilleures que d'autres.
C'est cette dernière source de variabilité qui nous intéresse. L'analyse par blocs est une manière de normaliser les erreurs de *leave-one-out* sur les instances $\langle x_i, y_i \rangle$. En se focalisant uniquement sur les rangs des

9. Dans ce contexte, $\{e^i_{loo}(\Lambda_1), \ldots, e^i_{loo}(\Lambda_k), \ldots, e^i_{loo}(\Lambda_K)\}$ est un exemple de bloc.

erreurs de *leave-one-out* dans chaque bloc, l'analyse par blocs élimine le risque que la variabilité entre les instances $\langle x_i, y_i \rangle$ cache la variabilité entre les classes d'hypothèses.

L'algorithme *F-race* sera comparé expérimentalement avec d'autres méthodes dans le chapitre 5. Dans cette thèse, nous allons utiliser la version de *F-race* implémentée dans la fonction *race* de la librairie homonyme [10] du langage de programmation R [11].

3.3 Le *multi-armed bandit problem*

Dans de nombreux problèmes, il est courant que des décisions doivent être prises en vue de maximiser une fonction de gain cumulé. Cette tâche est loin d'être triviale si l'environnement dans lequel les décisions doivent être prises est incertain. Dans ce contexte, il peut être utile de poser des actions qui n'ont pas pour but de maximiser le gain immédiat mais qui ont pour seul objectif d'améliorer la connaissance de l'environnement afin d'augmenter les gains à moyen terme.

Ces problèmes de maximisation d'une somme cumulée dans un environnement aléatoire peuvent être modélisés par le *multi-armed bandit problem* [11] (MABP). Dans ce problème, un joueur de casino est confronté à une machine à sous (en anglais *bandit machine*) qui a la particularité d'avoir K bras. Chaque bras a une probabilité de gain différente qui est, bien entendu, inconnue du joueur. Le joueur doit définir une stratégie séquentielle qui, à chaque étape, choisit le bras à jouer et ceci dans le but de maximiser les gains cumulés.

Cette section présente formellement le MABP et donne quelques exemples d'algorithmes qui le résolvent.

3.3.1 Formalisation du problème

Le MABP est défini par un ensemble de K variables aléatoires

$$\{\mathbf{z}_1, \ldots, \mathbf{z}_k, \ldots, \mathbf{z}_K\}.$$

Chaque variable aléatoire représente un des bras de la machine à sous. En fonction de la stratégie et sur base des gains déjà obtenus, le joueur sélectionne un bras à chaque étape l et ceci sur un total de H étapes. Le nombre de fois que l'alternative $\mathbf{z}_k$ a été testée durant les l premiers rounds est $\mathbf{n}_k$ et le vecteur $\mathbf{Z}_k(\mathbf{n}_k) = \left[\mathbf{z}_k^1, \mathbf{z}_k^2, \ldots, \mathbf{z}_k^{\mathbf{n}_k}\right]$ contient $\mathbf{n}_k$ variables aléatoires indépendantes et identiquement distribuées de moyenne μ_k inconnue. Dans cette thèse, nous nous limitons au cas où les variables sont

10. `http://cran.r-project.org/web/packages/race/index.html`
11. `http://www.r-project.org/`

stationnaires et indépendantes [12].

Le *regret de la variable k* est la différence entre le gain moyen espéré si le bras optimal est testé et le gain moyen espéré si le bras k est testé

$$\Delta_k = \mu_{[K]} - \mu_k.$$

L'ensemble $\mathbf{Z}_{\#} = \left\{\mathbf{Z}_1(\mathbf{n}_1^l), \ldots, \mathbf{Z}_K(\mathbf{n}_K^l)\right\}$ est un ensemble qui contient toutes les observations collectées. C'est sur base de ces observations qu'à chaque étape une stratégie π sélectionne dans $\{1, \ldots, K\}$ l'indice $\widehat{\mathbf{k}}$ de la variable à échantillonner. Un joueur utilisant π recevra un nouveau gain $\mathbf{z}_{\widehat{\mathbf{k}}}$. Le *regret à l'étape l d'une stratégie* π est

$$\delta_\pi^l = \mu_{[K]} - \mathbb{E}\left[\mathbf{z}_{\widehat{\mathbf{k}}}\right] \tag{3.10}$$

c'est-à-dire que δ_π^l est la différence entre le gain moyen si l'action optimale était appliquée et le gain moyen d'une action de la stratégie π à l'étape l.

Une stratégie optimale pour le MABP doit résoudre le compromis exploration/exploitation [50]. D'une part, comme les paramètres des distributions des variables aléatoires $\{\mathbf{z}_k\}$ sont inconnus, la stratégie doit faire un nombre de tests suffisants sur chaque alternative dans le but d'améliorer l'estimation des gains de chaque bras (*exploration*). D'autre part, comme le but est de maximiser la somme des gains, la stratégie doit aussi privilégier les tests sur l'alternative qui s'est montrée optimale jusqu'à présent (*exploitation*).

Pour mesurer la performance d'une stratégie qui résout le *multi-armed bandit problem*, le *regret cumulé d'une stratégie* π *sur* H *étapes*

$$\begin{aligned}
\rho_\pi^H &= H \cdot \mu_{[K]} - \sum_{k=1}^{K} \mathbb{E}\left[\mathbf{n}_k^H\right] \cdot \mu_k \\
&= \sum_{k=1}^{K} \mathbb{E}\left[\mathbf{n}_k^H\right] \cdot \Delta_k \\
&= \sum_{l=1}^{H} \delta_\pi^l
\end{aligned} \tag{3.11}$$

est typiquement considéré. Cette quantité ρ_π^H doit être minimisée. L'annexe C à la page 173 montre que les trois définitions du regret cumulé sont équivalentes.

Pour tout problème, une stratégie pour laquelle le regret cumulé par étape tend vers zéro lorsque H tend vers l'infini $\left(\lim_{H\to\infty} \frac{\rho^H}{H} = 0\right)$ est une stratégie à zéro-regret [86]. Intuitivement, une stratégie à zéro-regret converge vers une stratégie qui ne teste plus que l'alternative optimale.

12. [86] étudie le cas non-stationnaire et [73] étudie le cas où les variables sont dépendantes.

Il existe dans la littérature un grand nombre d'algorithmes pour résoudre le MABP. Une taxonomie est proposée dans [86] qui classifie dans des familles les principales stratégies existantes [13] :

- les stratégies semi-uniformes [87], caractérisées par une alternance de deux modes de fonctionnement qui sont le mode de pure exploitation et le mode de pure exploration ;
- les stratégies basées sur les intervalles de confiance, comme *UCB1* [6] et *β-UCB* [5], utilisant des bornes supérieures d'intervalles de confiance calculées sur les observations déjà obtenues ;
- les stratégies probabilistes, comme SoftMax [86] et Exp3 [7], qui choisissent une alternative à tester en fonction d'une distribution de probabilités mesurant la performance des bras ;
- les stratégies à index, comme l'index de Gittins [42], calculant – sur base d'une solution construite dans un cadre de programmation dynamique [12] – un index pour chaque alternative et testant l'alternative ayant le plus grand index.

Quelques-uns de ces algorithmes sont présentés dans la suite de cette section et comparés expérimentalement dans le chapitre 6 avec de nouveaux algorithmes proposés dans cette thèse.

3.3.2 Les stratégies semi-uniformes

Si le joueur estime, à chaque étape, le gain moyen de chaque bras et teste toujours le bras présentant le meilleur gain moyen estimé, alors le joueur adopte une stratégie gloutonne de pure exploitation. Cela signifie qu'il utilise sa connaissance sur les bras pour toujours sélectionner l'alternative semblant être l'optimum sans jamais explorer d'autres bras et ceci dans le but de maximiser le gain immédiat.

L'estimation du gain moyen de l'alternative $\mathbf{z}_k$ est la moyenne arithmétique des gains qui ont déjà été obtenus en testant cette alternative

$$\widehat{\boldsymbol{\mu}}_k = \frac{1}{\mathbf{n}_k} \sum_{i=1}^{\mathbf{n}_k} \mathbf{z}_k^i.$$

L'algorithme 8 décrit les étapes de la stratégie *greedy* [14]. Il faut d'abord tester chaque alternative au minimum une fois (ligne 1). Ensuite, pour toutes les autres étapes, l'algorithme adopte un comportement d'exploitation pure en sélectionnant (ligne 3) et en testant (ligne 4) toujours l'alternative ayant la plus grande moyenne arithmétique.

Si nous définissons par $\Pr\{S_k\}$ la probabilité qu'une alternative k soit testée par une action gloutonne où S_k correspond à l'événement « *sélection*

13. Il à est noter que les algorithmes commencent généralement par une petite phase d'exploration où chaque alternative est testée un petit nombre de fois.

14. Une action gloutonne est traduite en anglais par *greedy*.

Algorithme 8 La stratégie *greedy*

1: Tester au minimum une fois chaque alternative.
2: **boucler**
3: $\widehat{\mathbf{k}} \leftarrow \arg\max_k \{\widehat{\boldsymbol{\mu}}_k\}$.
4: Tester l'alternative $\widehat{\mathbf{k}}$.
5: **fin boucle**

de la variable $\mathbf{z}_k$ *comme ayant la plus grande moyenne arithmétique* $\gg$, alors le *regret à l'étape* l *d'une action gloutonne* est

$$\begin{aligned}
\delta_{greedy} &= \mu_{[K]} - \mathbb{E}\left[\mathbf{z}_{\widehat{\mathbf{k}}}\right] \\
&= \mu_{[K]} - \mathbb{E}\left[\mathbf{z}_{[\widehat{\mathbf{K}}]}\right] \\
&= \mu_{[K]} - \sum_{k=1}^{K} \Pr\{S_k\} \cdot \mathbb{E}[\mathbf{z}_k] \\
&= \mu_{[K]} - \sum_{k=1}^{K} \Pr\{S_k\} \cdot \mu_k \\
&= \mu_{[K]} - \mu_g^l
\end{aligned}$$

où dans le cas d'une action gloutonne $\widehat{\mathbf{k}} = [\widehat{\mathbf{K}}]$ et où μ_g^l est le gain moyen d'une action gloutonne à l'étape l. Une définition analytique de μ_g et de $\Pr\{S_k\}$ sera donnée comme contribution de la thèse dans le chapitre 4. Nous verrons que les définitions analytiques de ces deux quantités dépendent uniquement de $\{\mu_k\}$, $\{\sigma_k\}$ et $\{n_k\}$.

La nature de pure exploitation de la stratégie *greedy* ne lui permet pas d'être efficace lorsque le nombre d'alternatives K est élevé ou lorsque la variance des alternatives est trop grande. Il est par exemple possible que les premières fois que la meilleure alternative $[K]$ soit testée, celle-ci renvoie $\ll$ *par malchance* $\gg$ de mauvais gains. Elle sera alors considérée par l'algorithme *greedy* comme une mauvaise alternative et risque de ne plus jamais être testée. Pour résoudre cette difficulté, il faut que l'algorithme consacre une partie de son temps à l'exploration des autres alternatives.

Une variante naturelle de la stratégie *greedy* a été proposée par Watkins dans [87] ; elle consiste à garder une petite fraction du temps, définie par une quantité ϵ, pour effectuer des actions exploratoires. Cette stratégie, appelée ϵ-*greedy*, fait partie de la famille des stratégies *semi-uniformes* [86]. Les stratégies appartenant à cette famille se caractérisent par l'alternance de deux modes de fonctionnement bien distincts : un mode exploration et un mode exploitation. En mode exploration, la stratégie ϵ-*greedy* choisit aléatoirement le bras à tester, tandis qu'en mode exploitation c'est l'alternative ayant la plus grande moyenne arithmétique qui est testée.

Le paramètre $\epsilon \in [0, 1]$ est fixé par l'utilisateur et représente la balance

Algorithme 9 La stratégie *ϵ-greedy*

1: Tester au minimum une fois chaque alternative.
2: **boucler**
3: **e** $\leftarrow U[0,1]$ #Distribution uniforme
4: **si e** $< \epsilon$ **alors**
5: $\widehat{\mathbf{k}} \leftarrow$ choisir un bras aléatoirement selon une distribution uniforme.
6: **sinon**
7: $\widehat{\mathbf{k}} \leftarrow \arg\max_k \{\widehat{\boldsymbol{\mu}}_k\}$.
8: **fin si**
9: Tester l'alternative $\widehat{\mathbf{k}}$.
10: **fin boucle**

exploration/exploitation de la stratégie. Si ϵ est fixé à zéro, l'algorithme aura un comportement purement d'exploitation et testera toujours le bras qui s'est, jusqu'à présent, révélé optimal. Par contre, si ϵ est fixé à un, l'algorithme ne procédera qu'à une exploration aléatoire sans tenir compte des observations déjà collectées. Bien que très naïve, l'approche *ϵ-greedy* est réputée comme difficile à battre [15] sur certains types de problème.

L'algorithme 9 décrit les étapes de la stratégie *ϵ-greedy*. Ici aussi, il faut commencer par tester chaque alternative au minimum une fois (ligne 1). Ensuite démarre la boucle principale. Une variable est d'abord échantillonnée dans un intervalle $[0,1]$ à partir d'une distribution uniforme (ligne 3). Si la variable est plus petite que ϵ, la stratégie explore les diverses alternatives en faisant une sélection aléatoire uniforme dans l'ensemble $\{1,\ldots,K\}$ (ligne 5) et si la valeur est plus grande que ϵ, alors c'est une action d'exploitation gloutonne qui est exécutée (ligne 7).

Le gain moyen d'une action *ϵ-greedy* est (voir l'équation 3.10)

$$\mathbb{E}[\mu_{\widehat{\mathbf{k}}}] = \left(\epsilon \cdot \mu_r + (1-\epsilon) \cdot \mu_g^l\right)$$

où

$$\mu_r = \frac{1}{K}\sum_{k=1}^{K} \mu_k.$$

Le raisonnement est le suivant : lors d'une action, l'algorithme a soit une probabilité ϵ de choisir aléatoirement une alternative selon une distribution uniforme, soit une probabilité $(1-\epsilon)$ de choisir l'alternative avec la plus grande moyenne arithmétique. Si l'algorithme choisit une alternative aléatoirement, le gain moyen de l'action est la moyenne des espérances des bras. S'il choisit l'alternative ayant la plus grande moyenne arithmétique, alors le gain moyen de l'action est μ_g^l. Une définition analytique de μ_g^l sera donnée dans le chapitre 4.

15. ≪ *One remarkable outcome of our experiments is that the most naive approach, the ϵ-greedy strategy, proves to be often hard to beat.* ≫ [86]

Le théorème suivant montre que le caractère fixe du paramètre ϵ cause une croissance linéaire du regret cumulé ρ^H et que ϵ-*greedy* n'est donc pas une stratégie à zéro-regret [16].

Théorème 3.1. *Pour $K > 1$ et pour toute distribution des variables $\mathbf{z}_k$, si la stratégie est ϵ-greedy alors le regret cumulé après H étapes est borné inférieurement par une fonction linéaire*

$$\rho^H \geq H \cdot \epsilon \cdot \left(\mu_{[K]} - \mu_r\right)$$

où $\mu_r = 1/K \sum_{k=1}^{K} \mu_k$ est la moyenne arithmétique des K espérances.

Démonstration. On définit respectivement μ_r et μ_g^l comme les gains moyens d'actions aléatoires et gloutonnes où $\mu_r = 1/K \sum_{k=1}^{K} \mu_k$. Il est à noter que μ_r est constant durant toutes les étapes de l'algorithme. Le regret cumulé après H étapes de la stratégie ϵ-*greedy* est (voir équation (3.11))

$$\begin{aligned}
\rho^H &= \sum_{l=1}^{H} \mu_{[K]} - \left(\epsilon \cdot \mu_r + (1-\epsilon) \cdot \mu_g^l\right) \\
&= \sum_{l=1}^{H} \epsilon \cdot \mu_{[K]} + (1-\epsilon) \cdot \mu_{[K]} - \epsilon \cdot \mu_r - (1-\epsilon) \cdot \mu_g^l \\
&= H \cdot \epsilon \cdot \left(\mu_{[K]} - \mu_r\right) + (1-\epsilon) \sum_{l=1}^{H} \left(\mu_{[K]} - \mu_g^l\right).
\end{aligned}$$

Comme $\left(\mu_{[K]} - \mu_g^l\right)$ n'est jamais négatif, on obtient

$$\rho^H \geq H \cdot \epsilon \cdot \left(\mu_{[K]} - \mu_r\right).$$

□

Si le joueur a suffisamment exploré les différentes alternatives de telle sorte qu'une action d'exploitation de type glouton a de très fortes chances de tester l'alternative optimale $\mathbf{z}_{[K]}$, alors poser une action d'exploration se révèle sous-optimal. Intuitivement, une stratégie semi-uniforme optimale devrait concentrer toute l'exploration au début du problème de *bandit* et devrait, lorsque suffisamment d'informations ont été collectées sur les différents bras, converger vers un mode de pure exploitation. Une évolution naturelle de la stratégie ϵ-*greedy* est la stratégie ϵ_l-*greedy* [24] où le taux d'exploration est élevé au début et décroît ensuite.

L'algorithme 10 décrit les différentes étapes de ϵ_l-*greedy*. Cet algorithme

16. Nous fournissons ici un développement d'un théorème non-développé dans [6] : ≪ Clearly, the constant exploration probability ϵ [of ϵ-*greedy*] causes a linear [...] growth in the regret. ≫

Algorithme 10 La stratégie ϵ_l-*greedy*

1: Tester au minimum une fois chaque alternative.
2: **boucler**
3: $\quad \mathbf{e} \leftarrow U[0,1]$ #Distribution uniforme
4: $\quad \epsilon = \min\left\{1, \frac{\epsilon_0}{l}\right\}$
5: $\quad$ **si** $\mathbf{e} < \epsilon$ **alors**
6: $\quad\quad \widehat{\mathbf{k}} \leftarrow$ choisir un bras aléatoirement selon une distribution uniforme.
7: $\quad$ **sinon**
8: $\quad\quad \widehat{\mathbf{k}} \leftarrow \arg\max_k \{\widehat{\boldsymbol{\mu}}_k\}$.
9: $\quad$ **fin si**
10: $\quad$ Tester l'alternative $\widehat{\mathbf{k}}$.
11: **fin boucle**

est très proche de l'algorithme ϵ-*greedy*. La différence se situe à la ligne 4. Ici la valeur de ϵ n'est plus fixe mais décroît avec l.

Sous certaines conditions concernant les distributions $\{\mathbf{z}_k\}$ [17] et la valeur de ϵ_0, Auer et al. dans [24] ont trouvé une borne supérieure décroissante sur la probabilité de tester une alternative sous-optimale. Cela signifie qu'après suffisamment d'étapes, un joueur adoptant une stratégie ϵ_l-*greedy* a de fortes chances de toujours tester l'alternative optimale.

3.3.3 Les stratégies basées sur les intervalles de confiance

Dans cette famille de stratégies, on attribue un score à chaque alternative. Ce score est une « estimation optimiste du gain espéré » calculée sur base de la borne supérieure d'un intervalle de confiance et c'est l'alternative possédant le plus grand score qui est testée.

Les alternatives ayant une grande variance ou celles qui ont été peu testées présentent un score qui surévalue fortement leur vraie espérance μ. Ces alternatives seront donc testées plus souvent en début de procédure. L'idée est qu'au fur et à mesure que les tests sur les variables sont réalisés, les bornes supérieures (et donc les scores) se rapprochent des vraies moyennes μ. Après l'obtention d'un nombre suffisant d'observations, la stratégie ne devrait donc plus tester que l'alternative optimale.

La stratégie UCB1 [6] est un exemple très populaire d'algorithme faisant partie de cette famille. Cet algorithme est basé sur la borne de Hoeffding (voir annexe E). Soit $\widehat{\boldsymbol{\mu}}_k$ la moyenne arithmétique des observations collectées sur l'alternative k durant les l premières étapes. Si l'on fait uniquement l'hypothèse que les variables aléatoires associées aux alternatives prennent leurs valeurs dans $[0,1]$ alors

$$\Pr\{\widehat{\boldsymbol{\mu}}_k + \epsilon \geq \mu_k\} \geq 1 - e^{-2n_k\epsilon^2}. \tag{3.12}$$

17. Les valeurs prises par les variables aléatoires doivent l'être dans un intervalle dont les bornes supérieures et inférieures sont connues.

La quantité $\widehat{\boldsymbol{\mu}}_k + \epsilon$ est donc une borne optimiste de μ_k et ceci avec une garantie d'au moins $1 - e^{-2n_k\epsilon^2}$. Si l'on fixe la partie droite de (3.12) comme égale à $1 - 1/l^4$ et que l'on met ϵ en évidence, alors on obtient

$$\epsilon = \sqrt{\frac{2\ln(l)}{n_k}}$$

et donc

$$\widehat{\boldsymbol{\mu}}_k + \sqrt{\frac{2\ln(l)}{n_k}}$$

est une borne supérieure dans laquelle la vraie moyenne μ_k tombe avec une très forte probabilité. La stratégie UCB1 consiste à tester – à chaque étape – l'alternative qui maximise cette borne supérieure.

Il a été montré dans [6] que le regret cumulé ρ de la stratégie UCB1 est borné supérieurement par la fonction

$$\left[8 \sum_{k:\mu_k<\mu_{[K]}} \left(\frac{\ln(l)}{\Delta_k}\right)\right] + \left(1 + \frac{\pi^2}{3}\right)\left(\sum_{k=1}^{K} \Delta_k\right) \tag{3.13}$$

qui croît de manière logarithmique avec l.

3.3.4 Les stratégies probabilistes

Les stratégies de cette famille sont des méthodes qui choisissent l'alternative à tester sur base d'une distribution de probabilités. Cette distribution de probabilités mesure à quel point l'alternative est proche de l'optimum. Plus l'espérance de gain d'une alternative est grande, plus cette alternative aura de chances d'être sélectionnée pour être échantillonnée.

La stratégie SoftMax [57] (également appelée Bolzmann Exploration), consiste à choisir une alternative aléatoirement selon une distribution de Gibbs. L'alternative k est choisie avec une probabilité $e^{\widehat{\boldsymbol{\mu}}_k/\tau} / \sum_{i=1}^{K} e^{\widehat{\boldsymbol{\mu}}_i/\tau}$ où $\tau \in \mathbb{R}^+$ est un paramètre appelé « température ». C'est à l'utilisateur de l'algorithme SoftMax que revient la tâche de définir la valeur de ce paramètre τ. Plus la température est élevée, plus l'exploration sera forte.

Cette stratégie SoftMax peut également être modifiée de la même manière que l'algorithme semi-uniforme *ϵ-greedy* l'a été en fixant dès le départ une forte température et en la faisant ensuite décroître graduellement au fil des étapes. La stratégie D-SoftMax [24] est identique à la stratégie SoftMax à l'exception de la température $\tau_l = \tau_0/l$ qui dépend du numéro de l'étape actuelle l. Le choix de la valeur de la température initiale τ_0 est laissé à l'utilisateur.

3.3.5 La stratégie d'indice de Gittins

Gittins [42] propose d'utiliser les méthodes de programmation dynamique (voir annexe D) pour résoudre le MABP. Soit $s^l = \langle \widehat{\mu}_k, \widehat{\sigma}_k \rangle$ l'état observé de l'alternative k à l'étape l.

Supposons maintenant que la machine n'a plus que deux alternatives : le bras k et un autre bras « *spécial* » qui renvoie un gain constant z_0. Le joueur connaît la valeur de z_0 et, si cela lui semble intéressant, peut décider de ne jouer que ce bras « *spécial* ».

Nous définissons maintenant la *somme optimale des gains futurs espérés* à partir de l'état s^l. Partant de l'état s^l, si le joueur joue de manière optimale, cette quantité correspond à la somme des gains que le joueur va gagner (à un facteur de réduction α près). La *somme optimale des gains futurs espérés* est

$$J^*(s^l) = \max \left[\frac{z_0}{1-\alpha} \ , \ \widehat{\mu}_k + \alpha \int J^*(s^{l+1}(z_k)) \cdot \Pr\Big\{ \mathbf{z}_k = z_k \mid s^l \Big\} \cdot dz_k \right] \tag{3.14}$$

où α $(0 < \alpha \leq 1)$ est un facteur de réduction, $\Pr\{\mathbf{z}_k = z_k \mid s^l\}$ est la probabilité, sous l'hypothèse de normalité et connaissant s^l, que le gain soit égal à z_k et $s^{l+1}(z_k)$ est le nouvel état de l'alternative k connaissant z_k. Il est à noter que cette équation dynamique est à horizon infini.

Dans l'équation (3.14), à chaque étape le joueur a deux possibilités : soit il décide de jouer le bras k, soit il décide de jouer le bras « *spécial* ». Si cela semble plus intéressant de jouer le bras « *spécial* », le joueur recevra un gain z_0 et il n'obtiendra pas de nouvelles informations concernant le bras k. A l'étape suivante, le système sera dans le même état ($s^l = s^{l+1}$), et le joueur décidera donc à nouveau de jouer le bras « *spécial* » et ainsi de suite ; seul le bras spécial sera encore testé. La somme optimale des gains futurs espérés sera donc $z_0 + \alpha z_0 + \alpha^2 z_0 + \ldots = z_0 \sum_{i=0}^{\infty} \alpha^i = z_0/(1-\alpha)$ (voir équation 3.14). Si le joueur décide plutôt de jouer l'alternative k, il doit alors estimer la somme optimale des gains futurs espérés qui est composée de deux termes. Le premier terme $\widehat{\mu}_k$ mesure le gain immédiat estimé si l'alternative k est testée et le deuxième terme mesure les gains futurs espérés (à un facteur de réduction α près) après que le bras k ait été testé.

L'index de Gittins est défini comme la valeur z_0 pour laquelle les deux expressions dans les crochets de (3.14) sont égales. Cet index correspond à la valeur de z_0 à laquelle le joueur reste indifférent pour choisir entre le bras k et le bras « *spécial* ». En se basant sur des propriétés des distributions normales, Gittins propose une méthode pour calculer les index. Soit $v_g(\alpha, n_k)$ l'index de Gittins lorsque la moyenne est nulle et que l'écart-type vaut un. Dans un problème de maximisation, l'index de Gittins de l'alternative k est

$$\mathbf{v}_k = \widehat{\boldsymbol{\mu}}_k + \widehat{\boldsymbol{\sigma}}_k \cdot v_g(\alpha, n_k)$$

où quelques valeurs de $v_g(\alpha, n_k)$ peuvent être trouvées à la page 218 de [42]. Il faut ensuite tester l'alternative qui maximise $\mathbf{v}_k$.

3.4 Conclusions

Ce chapitre a introduit deux types de problème de sélection dans un environnement aléatoire où différentes alternatives sont disponibles. Chaque alternative est associée à une variable aléatoire dont la distribution est supposée inconnue et lorsqu'une alternative est testée, celle-ci produit en retour une observation.

Le premier cas de figure (voir section 3.2 à la page 43) est un problème de pure exploration. Il se divise en deux phases bien distinctes : une phase de tests et une phase de sélection de l'optimum. Durant la phase de tests, il s'agit de définir une stratégie d'échantillonnage qui vise à maximiser la probabilité de sélectionner l'optimum en fin des tests.

Dans ce premier type de problème, nous avons exposé différents algorithmes proposés par deux communautés scientifiques distinctes (simulation et apprentissage artificiel). Ces algorithmes échantillonnent toujours les alternatives en fonction du critère de probabilité de faire une sélection correcte $\Pr\{S_{[K]}\}$.

Le deuxième cas de figure (voir section 3.3 à la page 58) est le *multi-armed bandit problem*. Ici, il s'agit de trouver une stratégie d'échantillonnage qui maximise la somme des observations collectées. C'est un problème nécessitant un juste compromis entre exploration et exploitation. L'exploration a comme but de collecter davantage d'information sur les différentes alternatives, tandis que l'exploitation utilise l'information déjà obtenue pour réaliser une sélection la plus optimale possible.

Dans le chapitre suivant, la quantité $\Pr\{S_{[K]}\}$ sera utilisée pour définir analytiquement le « gain espéré d'une action gloutonne » μ_g. Quelques propriétés de cette quantité seront également définies. De nouveaux algorithmes seront proposés dans les chapitres 5 et 6 concernant le problème de sélection de l'optimum avec un budget de tests (*selecting the best*) et le problème du *bandit*. Ces algorithmes seront construits à partir des propriétés définies dans le chapitre suivant.

Chapitre 4

Le gain espéré d'une action gloutonne – contributions

Dans le chapitre précédent, nous avons montré que les actions gloutonnes jouent un rôle important dans les problèmes de sélection survenant dans un environnement aléatoire.

Dans les problèmes de sélection de l'optimum avec un budget de tests limité (voir section 3.2), après la première phase durant laquelle les observations sont collectées, c'est un algorithme de type glouton qui est utilisé pour sélectionner l'alternative $[\widehat{\mathbf{K}}]$ présentant la plus grande moyenne arithmétique.

Dans le *multi-armed bandit problem* (MABP) (voir section 3.3), les algorithmes semi-uniformes peuvent réaliser deux types d'action. Ils peuvent s'orienter soit vers l'exploitation et sélectionner de manière gloutonne l'alternative $[\widehat{\mathbf{K}}]$, soit opter pour l'exploration et avoir pour but d'améliorer les gains d'exploitation futurs.

Ce chapitre traite de l'action gloutonne et étudie plus spécifiquement les propriétés de son gain espéré. Les résultats théoriques obtenus dans ce chapitre seront ensuite utilisés pour proposer de nouveaux algorithmes permettant de résoudre le problème de sélection de l'optimum en un minimum de tests (Chapitre 5) ainsi que le *multi-armed bandit problem* (Chapitre 6). Ces nouveaux algorithmes seront comparés expérimentalement avec des méthodes précédemment définies dans le chapitre 3.

Ce chapitre présente les contributions de cette thèse se rapportant à l'étude des propriétés du *gain espéré d'une action gloutonne*, dénoté μ_g. La structure en est la suivante : la section 4.1 propose une définition analytique de μ_g dans le cas gaussien. Cette définition a l'inconvénient de supposer que les moyennes μ_k et les écarts-types σ_k des alternatives sont connus. La section 4.2 réalise une étude numérique sur l'évolution de μ_g qui permettra de proposer de nouveaux algorithmes d'exploration qui chercheront à maximiser μ_g. La section 4.3 propose des méthodes Monte Carlo afin de

calculer numériquement la valeur de la *probabilité de faire une sélection correcte* $\Pr\{S_{[K]}\}$. Enfin, la section 4.4 clôture ce chapitre en proposant une série d'estimateurs de μ_g construits à partir des observations collectées.

4.1 Définition du gain espéré d'une action gloutonne

Soit un ensemble $\{\mathbf{z}_k\}_{k\in[1,K]}$ de K variables aléatoires $\mathbf{z}_k$ de moyenne μ_k et d'écart-type σ_k. La variable $\mathbf{z}_k^i$ est définie comme la i-ième observation de la k-ième alternative et $\mathbf{Z}_k(n_k) = \left[\mathbf{z}_k^1, \mathbf{z}_k^2, \ldots, \mathbf{z}_k^{n_k}\right]$ est défini comme un vecteur contenant n_k observations indépendantes et identiquement distribuées de $\mathbf{z}_k$.

Une action gloutonne utilise les observations dans l'ensemble $\{\mathbf{Z}_k(n_k)\}$ où $k \in [1, K]$ pour calculer les moyennes arithmétiques $\{\widehat{\boldsymbol{\mu}}_k\}_{k\in[1,K]}$ des K alternatives et sélectionne la variable $\mathbf{z}_{[\widehat{\mathbf{K}}]}$ ayant la plus grande moyenne arithmétique, c'est-à-dire

$$[\widehat{\mathbf{K}}] = \arg\max\{\widehat{\boldsymbol{\mu}}_k\}$$

où

$$\widehat{\boldsymbol{\mu}}_k = \frac{\sum_{i=1}^{n_k} \mathbf{z}_k^i}{n_k}.$$

C'est ce type d'algorithme qui a été utilisé dans le chapitre 3 aussi bien pour les stratégies qui résolvent le problème de sélection de l'optimum en un minimum de tests que dans les stratégies semi-uniformes qui résolvent le *multi-armed bandit problem*.

Nous supposons que $[\widehat{\mathbf{K}}]$ est bien défini, c'est-à-dire qu'*une seule* alternative présente une moyenne arithmétique égale à $\widehat{\boldsymbol{\mu}}_{[\widehat{\mathbf{K}}]}$. Cette supposition se justifie par le fait que, comme les distributions sont continues, la probabilité que deux alternatives (ou plus) possèdent la même moyenne arithmétique est nulle.

Nous définissons

$$\Pr\{S_k\} = \Pr\Big\{[\widehat{\mathbf{K}}] = k\Big\} \quad , \; k = 1, \ldots, K$$

comme la probabilité que l'événement S_k se réalise, où S_k correspond à la sélection de l'alternative k par une action gloutonne. Il est à noter que $\Pr\{S_{[K]}\}$ représente la probabilité de faire une sélection correcte car $[K]$ est l'indice de l'alternative optimale ($[K] = \arg\max\{\mu_k\}$).

Nous savons que si l'alternative k est sélectionnée, le gain moyen sera μ_k. Ceci nous permet de définir le gain moyen obtenu après une action de type glouton. De manière générale, comme les K événements $\{S_1, \ldots, S_K\}$

sont mutuellement exclusifs

$$\sum_{k=1}^{K} \Pr\{S_k\} = 1,$$

le gain espéré d'une action gloutonne est

$$\mu_g = \sum_{k=1}^{K} \Pr\{S_k\} \cdot \mu_k.$$

Il reste à définir de manière analytique la probabilité de sélection $\Pr\{S_k\}$. Nous allons montrer dans le théorème suivant qu'il est possible de définir une expression analytique de cette probabilité dans le cas où les distributions de gains des variables aléatoires suivent une loi de distribution normale.

Cette expression suppose que les quantités μ_k et σ_k sont connues, alors qu'en réalité ces quantités sont bien entendu non disponibles. La section 4.4 va donc abandonner cette hypothèse irréaliste et proposer des estimateurs de μ_g qui seront ensuite utilisés dans les algorithmes des chapitres 5 et 6.

Théorème 4.1. *Soit $\{\mathbf{z}_1, \ldots, \mathbf{z}_K\}$, un ensemble de K variables aléatoires normales de moyenne μ_k et d'écart-type σ_k ($\mathbf{z}_k \sim \mathcal{N}[\mu_k, \sigma_k]$). La quantité n_k est définie comme le nombre d'observations disponibles de la variable $\mathbf{z}_k$. Si la procédure de sélection est gloutonne ($\arg\max_{k \in [1 \ldots K]} \{\widehat{\boldsymbol{\mu}}_k\}$), alors la probabilité de sélectionner $\mathbf{z}_{\bar{k}}$ est*

$$Pr\{S_{\bar{k}}\} = Pr\big\{\, \widehat{\mathbf{r}}_1 > 0 \;,\; \ldots \;,\; \widehat{\mathbf{r}}_{\bar{k}-1} > 0 \;,\; \widehat{\mathbf{r}}_{\bar{k}+1} > 0 \;,\; \ldots \;,\; \widehat{\mathbf{r}}_K > 0 \,\big\} \quad (4.1)$$

où $\left[\widehat{\mathbf{r}}_1, \ldots, \widehat{\mathbf{r}}_{\bar{k}-1}, \widehat{\mathbf{r}}_{\bar{k}+1}, \ldots, \widehat{\mathbf{r}}_K\right]^T$ suit une distribution normale multivariée

$$\left[\widehat{\mathbf{r}}_1, \ldots, \widehat{\mathbf{r}}_{\bar{k}-1}, \widehat{\mathbf{r}}_{\bar{k}+1}, \ldots, \widehat{\mathbf{r}}_K\right]^T \sim \mathcal{N}[\Gamma, \Sigma]$$

de moyenne

$$\Gamma = \begin{pmatrix} \mu_{\bar{k}} - \mu_1 \\ \vdots \\ \mu_{\bar{k}} - \mu_{\bar{k}-1} \\ \mu_{\bar{k}} - \mu_{\bar{k}+1} \\ \vdots \\ \mu_{\bar{k}} - \mu_K \end{pmatrix},$$

et de matrice de covariance

$$\Sigma = \begin{pmatrix} \frac{\sigma_{\bar{k}}^2}{n_{\bar{k}}} + \frac{\sigma_1^2}{n_1} & \cdots & \frac{\sigma_{\bar{k}}^2}{n_{\bar{k}}} & \frac{\sigma_{\bar{k}}^2}{n_{\bar{k}}} & \cdots & \frac{\sigma_{\bar{k}}^2}{n_{\bar{k}}} \\ \vdots & \ddots & \vdots & \vdots & & \vdots \\ \frac{\sigma_{\bar{k}}^2}{n_{\bar{k}}} & \cdots & \frac{\sigma_{\bar{k}}^2}{n_{\bar{k}}} + \frac{\sigma_{\bar{k}-1}^2}{n_{\bar{k}-1}} & \frac{\sigma_{\bar{k}}^2}{n_{\bar{k}}} & \cdots & \frac{\sigma_{\bar{k}}^2}{n_{\bar{k}}} \\ \frac{\sigma_{\bar{k}}^2}{n_{\bar{k}}} & \cdots & \frac{\sigma_{\bar{k}}^2}{n_{\bar{k}}} & \frac{\sigma_{\bar{k}}^2}{n_{\bar{k}}} + \frac{\sigma_{\bar{k}+1}^2}{n_{\bar{k}+1}} & \cdots & \frac{\sigma_{\bar{k}}^2}{n_{\bar{k}}} \\ \vdots & & \vdots & \vdots & \ddots & \vdots \\ \frac{\sigma_{\bar{k}}^2}{n_{\bar{k}}} & \cdots & \frac{\sigma_{\bar{k}}^2}{n_{\bar{k}}} & \frac{\sigma_{\bar{k}}^2}{n_{\bar{k}}} & \cdots & \frac{\sigma_{\bar{k}}^2}{n_{\bar{k}}} + \frac{\sigma_K^2}{n_K} \end{pmatrix}. \quad (4.2)$$

Démonstration. La quantité $\Pr\{S_{\bar{k}}\}$ est la probabilité que $\widehat{\boldsymbol{\mu}}_{\bar{k}}$ corresponde au maximum de l'ensemble $\{\widehat{\boldsymbol{\mu}}_1, \ldots, \widehat{\boldsymbol{\mu}}_K\}$:

$$\begin{aligned} \Pr\{S_{\bar{k}}\} &= \Pr\Big\{[\widehat{\mathbf{K}}] = \bar{k}\Big\} \\ &= \Pr\Big\{\bar{k} = \arg\max_{k\in[1\ldots K]} \{\widehat{\boldsymbol{\mu}}_k\}\Big\} \\ &= \Pr\big\{\, \widehat{\boldsymbol{\mu}}_{\bar{k}} > \widehat{\boldsymbol{\mu}}_1 \,,\; \ldots,\; \widehat{\boldsymbol{\mu}}_{\bar{k}} > \widehat{\boldsymbol{\mu}}_{\bar{k}-1} \,,\; \widehat{\boldsymbol{\mu}}_{\bar{k}} > \widehat{\boldsymbol{\mu}}_{\bar{k}+1} \,,\; \ldots,\; \widehat{\boldsymbol{\mu}}_{\bar{k}} > \widehat{\boldsymbol{\mu}}_K \,\big\} \\ &= \Pr\big\{\, \widehat{\mathbf{r}}_1 > 0 \,,\; \ldots \,,\; \widehat{\mathbf{r}}_{\bar{k}-1} > 0 \,,\; \widehat{\mathbf{r}}_{\bar{k}+1} > 0 \,,\; \ldots \,,\; \widehat{\mathbf{r}}_K > 0 \,\big\}, \end{aligned}$$

où $\widehat{\mathbf{r}}_k = \widehat{\boldsymbol{\mu}}_{\bar{k}} - \widehat{\boldsymbol{\mu}}_k$. La quantité $\Pr\{S_{\bar{k}}\}$ correspond donc également à la probabilité que tous les éléments du vecteur $\big[\widehat{\mathbf{r}}_1, \ldots, \widehat{\mathbf{r}}_{\bar{k}-1}, \widehat{\mathbf{r}}_{\bar{k}+1}, \ldots, \widehat{\mathbf{r}}_K\big]^T$ soient positifs. Dans l'hypothèse de Gaussianité, ce vecteur suit une loi de distribution normale multivariée

$$\begin{pmatrix} \widehat{\mathbf{r}}_1 \\ \vdots \\ \widehat{\mathbf{r}}_{\bar{k}-1} \\ \widehat{\mathbf{r}}_{\bar{k}+1} \\ \vdots \\ \widehat{\mathbf{r}}_K \end{pmatrix} = \begin{pmatrix} \widehat{\boldsymbol{\mu}}_{\bar{k}} - \widehat{\boldsymbol{\mu}}_1 \\ \vdots \\ \widehat{\boldsymbol{\mu}}_{\bar{k}} - \widehat{\boldsymbol{\mu}}_{\bar{k}-1} \\ \widehat{\boldsymbol{\mu}}_{\bar{k}} - \widehat{\boldsymbol{\mu}}_{\bar{k}+1} \\ \vdots \\ \widehat{\boldsymbol{\mu}}_{\bar{k}} - \widehat{\boldsymbol{\mu}}_K \end{pmatrix} \sim \mathcal{N}\left[\Gamma, \Sigma\right],$$

de moyenne

$$\Gamma = \begin{pmatrix} \mu_{\bar{k}} - \mu_1 \\ \vdots \\ \mu_{\bar{k}} - \mu_{\bar{k}-1} \\ \mu_{\bar{k}} - \mu_{\bar{k}+1} \\ \vdots \\ \mu_{\bar{k}} - \mu_K \end{pmatrix},$$

et de matrice de covariance

$$\Sigma = \begin{pmatrix} \sigma^2_{\widehat{\mathbf{r}}_1,\widehat{\mathbf{r}}_1} & \cdots & \sigma^2_{\widehat{\mathbf{r}}_1,\widehat{\mathbf{r}}_{\bar{k}-1}} & \sigma^2_{\widehat{\mathbf{r}}_1,\widehat{\mathbf{r}}_{\bar{k}+1}} & \cdots & \sigma^2_{\widehat{\mathbf{r}}_1,\widehat{\mathbf{r}}_K} \\ \vdots & \ddots & \vdots & \vdots & & \vdots \\ \sigma^2_{\widehat{\mathbf{r}}_{\bar{k}-1},\widehat{\mathbf{r}}_1} & \cdots & \sigma^2_{\widehat{\mathbf{r}}_{\bar{k}-1},\widehat{\mathbf{r}}_{\bar{k}-1}} & \sigma^2_{\widehat{\mathbf{r}}_{\bar{k}-1},\widehat{\mathbf{r}}_{\bar{k}+1}} & \cdots & \sigma^2_{\widehat{\mathbf{r}}_{\bar{k}-1},\widehat{\mathbf{r}}_K} \\ \sigma^2_{\widehat{\mathbf{r}}_{\bar{k}+1},\widehat{\mathbf{r}}_1} & \cdots & \sigma^2_{\widehat{\mathbf{r}}_{\bar{k}+1},\widehat{\mathbf{r}}_{\bar{k}-1}} & \sigma^2_{\widehat{\mathbf{r}}_{\bar{k}+1},\widehat{\mathbf{r}}_{\bar{k}+1}} & \cdots & \sigma^2_{\widehat{\mathbf{r}}_{\bar{k}+1},\widehat{\mathbf{r}}_K} \\ \vdots & & \vdots & \vdots & \ddots & \vdots \\ \sigma^2_{\widehat{\mathbf{r}}_K,\widehat{\mathbf{r}}_1} & \cdots & \sigma^2_{\widehat{\mathbf{r}}_K,\widehat{\mathbf{r}}_{\bar{k}-1}} & \sigma^2_{\widehat{\mathbf{r}}_K,\widehat{\mathbf{r}}_{\bar{k}+1}} & \cdots & \sigma^2_{\widehat{\mathbf{r}}_K,\widehat{\mathbf{r}}_K} \end{pmatrix}.$$

Puisque $\widehat{\boldsymbol{\mu}}_i$ et $\widehat{\boldsymbol{\mu}}_j$ sont indépendants pour $i \neq j$ et comme $\mathrm{Var}\big[\widehat{\boldsymbol{\mu}}_j\big] = \sigma_j^2/n_j$

$$\sigma^2_{\widehat{\mathbf{r}}_j,\widehat{\mathbf{r}}_j} = \mathrm{Var}\big[\widehat{\boldsymbol{\mu}}_{\bar{k}} - \widehat{\boldsymbol{\mu}}_j\big] = \frac{\sigma^2_{\bar{k}}}{n_{\bar{k}}} + \frac{\sigma^2_j}{n_j},$$

et

$$\begin{aligned} \sigma^2_{\widehat{\mathbf{r}}_i,\widehat{\mathbf{r}}_j} &= \sigma^2_{\widehat{\mathbf{r}}_j,\widehat{\mathbf{r}}_i} = \mathrm{cov}\left[\widehat{\boldsymbol{\mu}}_{\bar{k}} - \widehat{\boldsymbol{\mu}}_i, \widehat{\boldsymbol{\mu}}_{\bar{k}} - \widehat{\boldsymbol{\mu}}_j\right] \\ &= \mathbb{E}\big[(\widehat{\boldsymbol{\mu}}_{\bar{k}} - \widehat{\boldsymbol{\mu}}_i - \mathbb{E}[\widehat{\boldsymbol{\mu}}_{\bar{k}} - \widehat{\boldsymbol{\mu}}_i]) \cdot (\widehat{\boldsymbol{\mu}}_{\bar{k}} - \widehat{\boldsymbol{\mu}}_j - \mathbb{E}\big[\widehat{\boldsymbol{\mu}}_{\bar{k}} - \widehat{\boldsymbol{\mu}}_j\big])\big] \\ &= \mathbb{E}\big[(\widehat{\boldsymbol{\mu}}_{\bar{k}})^2\big] - \mu^2_{\bar{k}} \\ &= \frac{\sigma^2_{\bar{k}}}{n_{\bar{k}}}. \end{aligned}$$

□

Le gain espéré d'une action gloutonne est donc une quantité qui est fonction des moyennes des alternatives $\{\mu_k\}_{k\in[1,K]}$, de leurs écarts-types $\{\sigma_k\}_{k\in[1,K]}$ et du nombre de fois que les alternatives ont été testées $\{n_k\}_{k\in[1,K]}$.

Exemple numérique

Nous allons illustrer par un exemple les idées discutées précédemment.

Soit un problème composé de quatre alternatives $(\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4)$ ayant respectivement pour moyenne les valeurs $0, 1, 2, 1$ et pour variance les valeurs $2, 4, 1, 3$. On suppose que les alternatives ont toutes été testées dix fois et l'on voudrait connaître la valeur de la probabilité que l'alternative numéro deux soit sélectionnée par un algorithme glouton ($\Pr\{S_2\}$).

Selon le théorème 4.1, le vecteur $[\widehat{\mathbf{r}}_1, \widehat{\mathbf{r}}_2, \widehat{\mathbf{r}}_3]^T$ suit une distribution normale multivariée de moyenne $\Gamma = [1-0, 1-2, 1-1]^T = [1, -1, 0]^T$ et de matrice de covariance

$$\Sigma = \begin{pmatrix} \frac{4}{10} + \frac{2}{10} & \frac{4}{10} & \frac{4}{10} \\ \frac{4}{10} & \frac{4}{10} + \frac{1}{10} & \frac{4}{10} \\ \frac{4}{10} & \frac{4}{10} & \frac{4}{10} + \frac{3}{10} \end{pmatrix} = \begin{pmatrix} 3/5 & 2/5 & 2/5 \\ 2/5 & 1/2 & 2/5 \\ 2/5 & 2/5 & 7/10 \end{pmatrix}.$$

La probabilité de sélectionner – via un algorithme glouton – la variable $\mathbf{z}_2$ est égale à la probabilité que les trois variables ($\widehat{\mathbf{r}}_1$, $\widehat{\mathbf{r}}_2$ et $\widehat{\mathbf{r}}_3$) soient positives

$$\Pr\{\widehat{\mathbf{r}}_1 > 0, \widehat{\mathbf{r}}_2 > 0, \widehat{\mathbf{r}}_3 > 0\}$$

et dans cet exemple, cette probabilité vaut 0.074. La section 4.3 étudiera des techniques de calcul des probabilités de normales multivariées.

4.1.1 Le gain espéré d'une action gloutonne lorsque $K = 2$

Soit $[K]$ et $[K-1]$, deux alternatives telles que $\mu_{[K]} > \mu_{[K-1]}$. Dans le cas où seules deux alternatives sont disponibles ($K = 2$), le gain d'une action gloutonne est

$$\mu_g = \Pr\{S_{[K]}\} \cdot \mu_{[K]} + \Pr\{S_{[K-1]}\} \cdot \mu_{[K-1]}$$

où la quantité $\Pr\{S_{[K]}\}$ est la probabilité d'opérer une sélection correcte avec un algorithme glouton et $\Pr\{S_{[K-1]}\}$ est la probabilité qu'une action gloutonne sélectionne la mauvaise alternative.

Il est à noter que, dans ce cas très simple ($K = 2$), la probabilité de faire une sélection correcte $\Pr\{S_{[K]}\}$ est la probabilité que $\widehat{\boldsymbol{\mu}}_{[K]} - \widehat{\boldsymbol{\mu}}_{[K-1]}$ soit positif, où $\widehat{\boldsymbol{\mu}}_{[K]} - \widehat{\boldsymbol{\mu}}_{[K-1]}$ possède une distribution normale de moyenne $\mu_{[K]} - \mu_{[K-1]}$ et d'écart-type $\sqrt{\frac{\sigma^2_{[K]}}{n_{[K]}} + \frac{\sigma^2_{[K-1]}}{n_{[K-1]}}}$ (voir figure 4.1).

Le théorème 4.4 à la page 80 va utiliser ce fait pour étudier les propriétés d'un algorithme d'exploration qui – dans le cas où $K = 2$ – sélectionne toujours l'alternative maximisant $\Pr\{S_{[K]}\}$ à l'étape suivante.

4.1.2 Le gain espéré d'une action gloutonne dans la sélection de modèles

Considérons maintenant la sélection de modèles dans les problèmes de régression en apprentissage artificiel (voir chapitre 2). Nous disposons d'un ensemble $\mathbf{D}_N$ contenant des exemples bruités d'un phénomène d'entrée/sortie $f(\cdot)$. Nous supposons que l'hypothèse $h(\cdot, \boldsymbol{\alpha}_N^k)$ a été choisie via un algorithme d'identification paramétrique $\mathcal{I}^P$ qui a cherché $\boldsymbol{\alpha}_N^k$ dans un ensemble Λ_k à partir de $\mathbf{D}_N$

$$\boldsymbol{\alpha}_N^k = \mathcal{I}^P\left(\Lambda_k, \mathbf{D}_N\right).$$

Partant d'une classe d'hypothèses Λ_k sur laquelle un algorithme $\mathcal{I}^P$ est appliqué, l'erreur de généralisation MISE (*mean integrated squared error*) est (voir équation (2.9) à la page 29)

$$\text{MISE}(\Lambda_k) = \mathbb{E}[\mathbf{e} \mid \Lambda_k]$$

où $\mathbf{e} = \left(\mathbf{y} - h\left(\mathbf{x}, \mathcal{I}^P\left(\Lambda_k, \mathbf{D}_N\right)\right)\right)^2$.

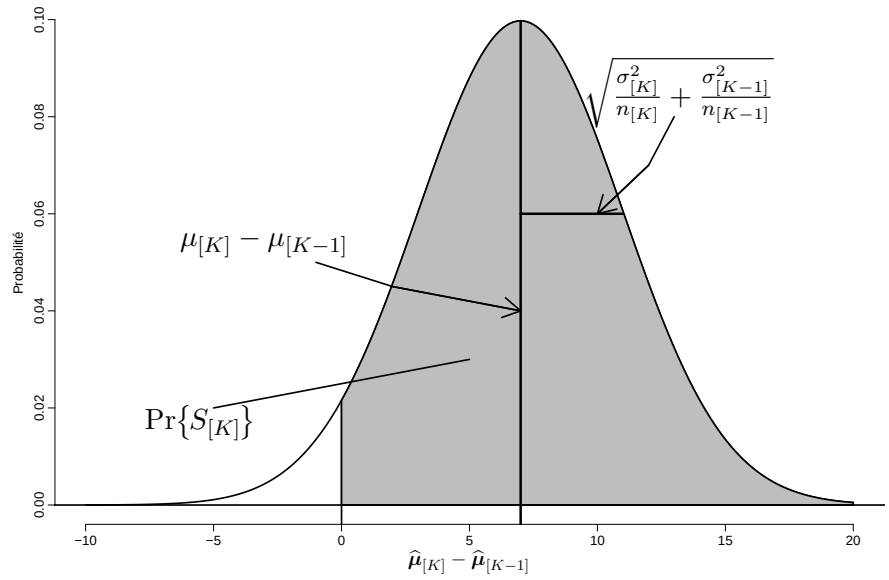


FIGURE 4.1 – Si $K = 2$, la probabilité de faire une sélection correcte avec une action gloutonne est égale à la surface sous la Gaussienne lorsque l'abscisse prend ses valeurs dans l'intervalle $[0, +\inf]$. Il est à noter que, par définition, la moyenne de cette Gaussienne est toujours positive.

Cette erreur de généralisation demeure inaccessible et doit être estimée. Ceci peut s'effectuer via la méthode d'estimation par *leave-one-out* (voir section 2.4.3) qui consiste à calculer une série de N erreurs de *leave-one-out* $\{\mathbf{e}^1_{loo}, \ldots, \mathbf{e}^i_{loo}, \ldots, \mathbf{e}^N_{loo}\}$. La moyenne arithmétique de ces erreurs équivaut alors à une estimation de la MISE.

Jusqu'à présent, nous avons supposé que les variables aléatoires $\mathbf{z}_k$ suivent des distributions normales. Comme les erreurs de *leave-one-out* sont quadratiques, cette hypothèse n'est plus vérifiée. Considérons alors que la différence entre l'hypothèse $h(\mathbf{x}_i, \boldsymbol{\alpha}^k_{N(i)})$ et $\mathbf{y}_i$ suit une loi de distribution normale où $\boldsymbol{\alpha}^k_{N(i)}$ comporte les caractéristiques de l'hypothèse construite par $\mathcal{I}^P$ à partir de l'ensemble $\mathbf{D}_N$ duquel $\langle \mathbf{x}_i, \mathbf{y}_i \rangle$ a été retiré (voir section 2.4.3). Dans ce cas, les erreurs de *leave-one-out* sont des carrés de variable normale

$$\begin{aligned} \mathbf{e}^i_{loo} &= \left(\mathbf{y}_i - h(\mathbf{x}_i, \boldsymbol{\alpha}^k_{N(i)})\right)^2 \\ &= (\mathbf{v}_i)^2 \end{aligned}$$

où $\mathbf{v}_i$ est défini comme une variable aléatoire normale de moyenne μ_v et d'écart-type σ_v. Il est à noter que μ_v représente le biais de la classe d'hypothèses Λ_k si $N - 1$ exemples sont utilisés pour l'apprentissage [1].

1. Lors de l'estimation par *leave-one-out*, l'identification paramétrique est réalisée avec

Dans cette section, nous allons définir la quantité μ_g dans le contexte de l'apprentissage artificiel. La somme des N erreurs de *leave-one-out* est

$$\begin{aligned}\sum_{i=1}^{N} \mathbf{e}_{loo}^{i} &= \sum_{i=1}^{N} (\mathbf{v}_i)^2 \\ &= \sum_{i=1}^{N} (\sigma_v \cdot \mathbf{p} + \mu_v)^2 \\ &= \sum_{i=1}^{N} \left(\sigma_v^2 \cdot \mathbf{p}^2 + \mu_v^2 + 2 \cdot \sigma_v \cdot \mu_v \cdot \mathbf{p}\right) \\ &= \sigma_v^2 \cdot \mathbf{E} + N \cdot \mu_v^2 + 2 \cdot \sigma_v \cdot \mu_v \cdot \mathbf{L}\end{aligned}$$

où $\mathbf{p}$ est une variable aléatoire normale de moyenne nulle et d'écart-type un, $\mathbf{E}$ est une variable aléatoire qui suit une distribution khi-carrée à N degrés de liberté (voir annexe F) et enfin $\mathbf{L}$ est une variable normale de moyenne nulle et d'écart-type $\sqrt{N}$. La moyenne arithmétique des N erreurs de *leave-one-out* est donc

$$\widehat{\mathbf{MISE}}_{loo}(\Lambda_k) = \frac{\sum_{i=1}^{N} \mathbf{e}_{loo}^{i}}{N} = \frac{\sigma_v^2 \cdot \mathbf{E}}{N} + \mu_v^2 + \frac{2 \cdot \sigma_v \cdot \mu_v \cdot \mathbf{L}}{N} \tag{4.3}$$

qui est une estimation du MISE(Λ_k) et sa moyenne est de

$$\begin{aligned}\mathbb{E}\left[\widehat{\mathbf{MISE}}_{loo}(\Lambda_k)\right] &= \frac{\sigma_v^2 \cdot \mathbb{E}[\mathbf{E}]}{N} + \mu_v^2 + \frac{2 \cdot \sigma_v \cdot \mu_v \cdot \mathbb{E}[\mathbf{L}]}{N} \\ &= \mu_v^2 + \sigma_v^2.\end{aligned}$$

Si une série de classes d'hypothèses est en compétition, une série d'estimations

$$\left\{\widehat{\mathbf{MISE}}_{loo}(\Lambda_1), \ldots, \widehat{\mathbf{MISE}}_{loo}(\Lambda_k), \ldots, \widehat{\mathbf{MISE}}_{loo}(\Lambda_K)\right\}$$

peut alors être calculée. Une approche classique, appelée en apprentissage artificiel *winner takes all* [17], consiste à choisir de manière gloutonne la classe Λ qui *minimise* $\widehat{\mathbf{MISE}}_{loo}$.

A partir d'un ensemble de K estimations, la quantité

$$\Pr\{S_k^-\} = \Pr\left\{k = \arg\min \widehat{\mathbf{MISE}}_{loo}(\Lambda_k)\right\}$$

est définie comme la probabilité que la classe d'hypothèses Λ_k possède la *plus petite* estimation de l'erreur de généralisation. Nous savons que si cette classe d'hypothèses Λ_k est sélectionnée, son erreur de généralisation équivaut

$N-1$ observations.

à MISE_k. Ceci nous permet de définir l'erreur de généralisation moyenne obtenue en appliquant l'approche *winner takes all*. De manière générale, comme les K événements $\{S_1^-, \ldots, S_K^-\}$ sont mutuellement exclusifs

$$\sum_{k=1}^{K} \Pr\{S_k^-\} = 1,$$

l'erreur de généralisation moyenne obtenue en appliquant l'approche *winner takes all* est

$$\text{MISE}_g = \sum \Pr\{S_k^-\} \cdot \text{MISE}_k.$$

Cette quantité MISE_g correspond à la quantité μ_g dans le cas des problèmes d'apprentissage artificiel.

4.2 Etude sur l'évolution de la quantité μ_g

Une difficulté majeure dans les problèmes d'optimisation étudiés ici réside dans la dynamique et la nature multivariée des termes entrant en compte dans une stratégie optimale. Vu l'importance des actions de type glouton, il est intéressant d'étudier l'évolution – au cours du temps – du gain espéré d'une telle action. Ceci fera l'objet de la présente section.

L'objectif d'une sélection exploratoire peut être de maximiser la probabilité de faire une sélection correcte $\Pr\{S_{[K]}\}$. Comme μ_g atteint son maximum lorsque $\Pr\{S_{[K]}\} = 1$, optimiser μ_g est une manière indirecte d'optimiser $\Pr\{S_{[K]}\}$. Les conclusions tirées quant à l'évolution de μ_g permettront donc de proposer de nouveaux algorithmes d'exploration.

Cette section se divise en deux parties. Dans la première partie, nous supposons que seules deux alternatives sont disponibles à savoir $\mathbf{z}_{[K]}$, l'alternative optimale et $\mathbf{z}_{[K-1]}$, l'autre alternative $\left(\mu_{[K]} > \mu_{[K-1]}\right)$. Il s'agit du cas le plus simple dans lequel quelques résultats analytiques concernant l'évolution de μ_g vont pouvoir être obtenus. La deuxième partie étudie le cas plus général où $K > 2$. Ici, il n'a pas été possible de réaliser une étude analytique sur l'évolution de μ_g mais des résultats intéressants ont pu être observés de manière empirique.

4.2.1 L'évolution de μ_g lorsque $K = 2$

Dans un premier temps, nous considérons que deux alternatives sont disponibles. Le théorème 4.2 montre que si deux alternatives sont en compétition ($K = 2$), alors tester $\mathbf{z}_{[K]}$ ou $\mathbf{z}_{[K-1]}$ à l'étape courante (l) augmente, dans les deux cas, la valeur de μ_g à l'étape suivante ($l + 1$). Cette conclusion ne sera plus systématiquement observée lorsque plus de deux alternatives seront en compétition ($K > 2$). Le théorème 4.3 montre que, pour une stratégie testant toujours la même variable, l'évolution de μ_g converge vers une borne

supérieure b^{∞} d'une valeur plus petite que $\mu_{[K]}$. Ceci constitue un argument plaidant en faveur de la nécessité de débuter toute prise de décision séquentielle par une phase d'exploration. Enfin, le théorème 4.4 étudie les situations d'exploration optimale qui maximisent μ_g à l'étape $l+1$.

Théorème 4.2. *Soit un problème où $K = 2$ alternatives dont les variables suivent des distributions normales et où l'on définit $\mu_g^{l+1}(k)$ comme la valeur de μ_g à l'étape $l+1$ si l'alternative k est testée à l'étape l. On peut alors montrer que*

$$\forall k \in \{1,2\},\ \mu_g^{l+1}(k) > \mu_g^l.$$

Démonstration. $\forall k \in \{1,2\}, \mu_g^{l+1}(k) > \mu_g^l$ si et seulement si

$$\forall k \in \{1,2\}, \Pr\left\{S_{[K]}^{l+1}(k)\right\} > \Pr\left\{S_{[K]}\right\}$$

où $\Pr\left\{S_{[K]}^{l+1}(k)\right\}$ est la probabilité de faire une sélection correcte via une action gloutonne à l'étape $l+1$ si l'alternative $\mathbf{z}_k$ est testée à l'étape l. Par définition

$$\Pr\left\{S_{[K]}\right\} = \Pr\left\{\widehat{\boldsymbol{\mu}}_{[K]} - \widehat{\boldsymbol{\mu}}_{[K-1]} > 0\right\} \tag{4.4}$$

et sous l'hypothèse de normalité, nous avons

$$\left(\widehat{\boldsymbol{\mu}}_{[K]} - \widehat{\boldsymbol{\mu}}_{[K-1]}\right) \sim \mathcal{N}\left(\mu_{[K]} - \mu_{[K-1]}, \frac{\sigma_{[K]}^2}{n_{[K]}} + \frac{\sigma_{[K-1]}^2}{n_{[K-1]}}\right).$$

L'équation (4.4) peut être réécrite comme suit

$$\begin{aligned}\Pr\left\{S_{[K]}\right\} &= 1 - \Pr\left\{\widehat{\boldsymbol{\mu}}_{[K]} - \widehat{\boldsymbol{\mu}}_{[K-1]} \leq 0\right\} \\ &= 1 - \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{\mu_{[K-1]} - \mu_{[K]}}{\left(\frac{\sigma_{[K]}^2}{n_{[K]}} + \frac{\sigma_{[K-1]}^2}{n_{[K-1]}}\right)\sqrt{2}}\right)\right]\end{aligned}$$

où $\operatorname{erf}(\cdot)$ est la fonction d'erreur de Gauss [83]. Tester $\mathbf{z}_{[K]}$ ou $\mathbf{z}_{[K-1]}$ augmentera respectivement soit $n_{[K]}$ soit $n_{[K-1]}$. Comme $\mu_{[K-1]} - \mu_{[K]}$ est, par définition, négatif et que la dérivée de la fonction $\operatorname{erf}(\cdot)$ est toujours positive (c'est-à-dire que $\operatorname{erf}(\cdot)$ est une fonction croissante), tester l'une des deux alternatives fera – dans tous les cas – augmenter la probabilité $\Pr\left\{S_{[K]}\right\}$ de faire une sélection correcte. □

Nous sommes donc certains que, quelle que soit l'alternative testée, les gains moyens futurs d'une action gloutonne ne pourront qu'augmenter. Cette croissance est néanmoins bornée supérieurement. Cette borne peut au maximum équivaloir à $\mu_{[K]}$ mais nous verrons dans le théorème suivant qu'elle sera moins élevée dans le cas d'une stratégie testant toujours la même alternative.

Théorème 4.3. *Soit $\mathbf{z}_{k_1}$ et $\mathbf{z}_{k_2}$, deux variables aléatoires distribuées selon une normale. Supposons que l'on adopte une stratégie qui teste toujours la même alternative $\mathbf{z}_{k_1}$ tandis que $\mathbf{z}_{k_2}$, l'autre alternative, n'est jamais testée. Si*

$$\mathbf{X} \sim \mathcal{N}\left(\mu_{k_1} - \mu_{k_2}, \frac{\sigma_{k_2}^2}{n_{k_2}}\right)$$

alors

$$\forall l \geq 1 \ , \quad \mu_g^l < b^\infty < \mu_{[K]},$$

où

$$b^\infty = (\mu_{k_1} - \mu_{k_2}) \cdot Pr\{\mathbf{X} > 0\} + \mu_{k_2}.$$

Démonstration. Soit τ, un entier positif. Nous pouvons établir une borne supérieure pour μ_g^l (voir théorème 4.2)

$$\begin{aligned} \mu_g^l &< \mu_g^{l+1}(k_1) \\ &\leq \mu_g^{l+\tau}(k_1) \end{aligned}$$

où $\mu_g^{l+\tau}(k_1)$ est le futur gain espéré d'une action gloutonne lorsque l'alternative k_1 est testée τ fois d'affilée à partir de l'itération l avec $l = n_{k_1} + n_{k_2}$. Cette quantité est définie analytiquement par

$$\begin{aligned} \mu_g^{l+\tau}(k_1) &= \mu_{k_1} \cdot \Pr\left\{\left(\widehat{\boldsymbol{\mu}}_{k_1} - \widehat{\boldsymbol{\mu}}_{k_2}\right)_\tau > 0\right\} + \mu_{k_2} \cdot \left(1 - \Pr\left\{\left(\widehat{\boldsymbol{\mu}}_{k_1} - \widehat{\boldsymbol{\mu}}_{k_2}\right)_\tau > 0\right\}\right) \\ &= (\mu_{k_1} - \mu_{k_2}) \cdot \Pr\left\{\left(\widehat{\boldsymbol{\mu}}_{k_1} - \widehat{\boldsymbol{\mu}}_{k_2}\right)_\tau > 0\right\} + \mu_{k_2} \qquad (4.5) \\ &= b^\tau \end{aligned}$$

où

$$\left(\widehat{\boldsymbol{\mu}}_{k_1} - \widehat{\boldsymbol{\mu}}_{k_2}\right)_\tau \sim \mathcal{N}\left(\mu_{k_1} - \mu_{k_2}, Var_\tau\right)$$

et où

$$Var_\tau = \frac{\sigma_{k_1}^2}{n_{k_1} + \tau} + \frac{\sigma_{k_2}^2}{n_{k_2}}.$$

Si la stratégie utilise de façon infinie l'alternative k_1 sans jamais tester l'alternative k_2, alors la variance de $\left(\widehat{\boldsymbol{\mu}}_{k_1} - \widehat{\boldsymbol{\mu}}_{k_2}\right)_\tau$ converge vers

$$\lim_{\tau \to \infty} Var_\tau = \lim_{\tau \to \infty} \left(\frac{\sigma_{k_1}^2}{n_{k_1} + \tau} + \frac{\sigma_{k_2}^2}{n_{k_2}}\right) = \frac{\sigma_{k_2}^2}{n_{k_2}}.$$

Soit $\mathbf{X}$, une variable définie comme suit

$$\mathbf{X} = \left(\widehat{\boldsymbol{\mu}}_{k_1} - \widehat{\boldsymbol{\mu}}_{k_2}\right)_\infty \sim \mathcal{N}\left(\mu_{k_1} - \mu_{k_2}, \frac{\sigma_{k_2}^2}{n_{k_2}}\right). \qquad (4.6)$$

A partir des équations (4.5) et (4.6), une borne supérieure de μ_g^l est fournie par

$$b^\infty = (\mu_{k_1} - \mu_{k_2}) \cdot \Pr\{\mathbf{X} > 0\} + \mu_{k_2}$$

et nous avons $\mu_g < b^{\infty}$.

Il reste à démontrer que $b^{\infty} < \mu_{[K]}$. Cette borne supérieure b^{∞} vaut $\mu_{[K]}$ seulement si $\Pr\{S_{[K]}\} = 1$. Or $\Pr\{S_{[K]}\}$ vaut un uniquement lorsque la variance $\frac{\sigma^2_{k_1}}{n_{k_1}} + \frac{\sigma^2_{k_2}}{n_{k_2}} = 0$. Dans le cas d'une stratégie qui ne teste jamais l'alternative k_2, la variance $\frac{\sigma^2_{k_1}}{n_{k_1}} + \frac{\sigma^2_{k_2}}{n_{k_2}}$ converge vers $\frac{\sigma^2_{k_2}}{n_{k_2}}$ qui est supérieur à zéro. Comme cette variance n'atteint pas zéro, $\Pr\{S_{[K]}\}$ ne vaut pas un et nous avons donc $b^{\infty} < \mu_{[K]}$. □

Tester l'une ou l'autre alternative n'est donc pas équivalent. Le théorème suivant étudie les situations d'exploration optimale et sera par la suite utilisé afin de proposer un algorithme d'exploration, dénoté optiK2.

Théorème 4.4. *Soit un problème de sélection comportant deux variables aléatoires* $\{\mathbf{z}_{k_1}, \mathbf{z}_{k_2}\}$ *distribuées selon des lois de distribution normale. La sélection*

$$\begin{cases} \textit{si } N_{\Delta} < 0 & \textit{alors tester } \mathbf{z}_{k_1} \\ \textit{si } N_{\Delta} > 0 & \textit{alors tester } \mathbf{z}_{k_2} \\ \textit{si } N_{\Delta} = 0 & \textit{alors tester indifféremment } \mathbf{z}_{k_1} \textit{ ou } \mathbf{z}_{k_2} \end{cases}$$

où

$$N_{\Delta} = n_{k_1} \cdot (n_{k_1} + 1) \cdot \left(\sigma^2_{k_2} - \sigma^2_{k_1}\right) + \sigma^2_{k_1} \cdot (n_{k_2} + n_{k_1} + 1) \cdot (n_{k_1} - n_{k_2}) \quad (4.7)$$

maximise la valeur de μ_g *à l'étape* $l + 1$.

Démonstration. Si $\left[\mu_g^{l+1}(k_1) - \mu_g^{l+1}(k_2)\right]$ est positif alors $\mathbf{z}_{k_1}$ doit être testé pour maximiser μ_g à l'étape $l + 1$ et si $\left[\mu_g^{l+1}(k_1) - \mu_g^{l+1}(k_2)\right]$ est négatif alors c'est $\mathbf{z}_{k_2}$ qui doit être testé. Si $\mu_g^{l+1}(k_1)$ et $\mu_g^{l+1}(k_2)$ sont égaux alors tester $\mathbf{z}_{k_1}$ ou $\mathbf{z}_{k_2}$ donne le même résultat quant à l'évolution de μ_g.

$$\begin{aligned}
&\mu_g^{l+1}(k_1) - \mu_g^{l+1}(k_2) \\
&= \Pr\left\{S_{[K]}^{l+1}(k_1)\right\} \cdot \mu_{[K]} + \Pr\left\{S_{[K-1]}^{l+1}(k_1)\right\} \cdot \mu_{[K-1]} \\
&- \Pr\left\{S_{[K]}^{l+1}(k_2)\right\} \cdot \mu_{[K]} - \Pr\left\{S_{[K-1]}^{l+1}(k_2)\right\} \cdot \mu_{[K-1]} \\
&= \left[\Pr\left\{S_{[K]}^{l+1}(k_1)\right\} - \Pr\left\{S_{[K]}^{l+1}(k_2)\right\}\right] \cdot \mu_{[K]} \\
&+ \left[\Pr\left\{S_{[K-1]}^{l+1}(k_1)\right\} - \Pr\left\{S_{[K-1]}^{l+1}(k_2)\right\}\right] \cdot \mu_{[K-1]} \\
&= \left[\Pr\left\{S_{[K]}^{l+1}(k_1)\right\} - \Pr\left\{S_{[K]}^{l+1}(k_2)\right\}\right] \cdot \mu_{[K]} \\
&- \left[\Pr\left\{S_{[K]}^{l+1}(k_1)\right\} - \Pr\left\{S_{[K]}^{l+1}(k_2)\right\}\right] \cdot \mu_{[K-1]} \\
&= \left[\Pr\left\{S_{[K]}^{l+1}(k_1)\right\} - \Pr\left\{S_{[K]}^{l+1}(k_2)\right\}\right] \cdot \left(\mu_{[K]} - \mu_{[K-1]}\right) \\
&= \Delta P_{[K]} \cdot \left(\mu_{[K]} - \mu_{[K-1]}\right).
\end{aligned}$$

Comme $\left(\mu_{[K]} - \mu_{[K-1]}\right) > 0$, le signe de $\left[\mu_g^{l+1}(k_1) - \mu_g^{l+1}(k_2)\right]$ est identique à celui de $\Delta P_{[K]}$.

Soit $\mathrm{Var}^{k_1}\left(\widehat{\boldsymbol{\mu}}_{[K]}^{l+1} - \widehat{\boldsymbol{\mu}}_{[K-1]}^{l+1}\right)$ et $\mathrm{Var}^{k_2}\left(\widehat{\boldsymbol{\mu}}_{[K]}^{l+1} - \widehat{\boldsymbol{\mu}}_{[K-1]}^{l+1}\right)$, respectivement les variances de $\left(\widehat{\boldsymbol{\mu}}_{[K]}^{l+1} - \widehat{\boldsymbol{\mu}}_{[K-1]}^{l+1}\right)$ à l'étape $l+1$ si $\mathbf{z}_{k_1}$ ou $\mathbf{z}_{k_2}$ est testé à l'étape courante l. $\Delta\mathrm{Var}$ est défini comme la différence des deux variances à l'étape $l+1$. Réduire la variance de $\left(\widehat{\boldsymbol{\mu}}_{[K]}^{l+1} - \widehat{\boldsymbol{\mu}}_{[K-1]}^{l+1}\right)$ augmente la probabilité $\Pr\{S_{[K]}\}$ de faire une sélection correcte. Le signe de $\Delta\mathrm{Var}$ est donc l'opposé du signe de $\Delta P_{[K]}$. La quantité $\Delta\mathrm{Var}$ peut se réécrire comme suit

$$\begin{aligned}
\Delta\mathrm{Var} &= \mathrm{Var}^{k_1}\left(\widehat{\boldsymbol{\mu}}_{[K]}^{l+1} - \widehat{\boldsymbol{\mu}}_{[K-1]}^{l+1}\right) - \mathrm{Var}^{k_2}\left(\widehat{\boldsymbol{\mu}}_{[K]}^{l+1} - \widehat{\boldsymbol{\mu}}_{[K-1]}^{l+1}\right) \\
&= \frac{\sigma_{k_1}^2}{n_{k_1}+1} + \frac{\sigma_{k_2}^2}{n_{k_2}} - \frac{\sigma_{k_1}^2}{n_{k_1}} - \frac{\sigma_{k_2}^2}{n_{k_2}+1} \\
&= \frac{N_\Delta}{D_\Delta}
\end{aligned}$$

où

$$\begin{aligned}
N_\Delta &= n_{k_2} n_{k_1}\left(n_{k_2}+1\right)\sigma_{k_1}^2 + \left(n_{k_1}+1\right) n_{k_1}\left(n_{k_2}+1\right)\sigma_{k_2}^2 \\
&- \left(n_{k_1}+1\right) n_{k_2}\left(n_{k_2}+1\right)\sigma_{k_1}^2 - \left(n_{k_1}+1\right) n_{k_2} n_{k_1}\sigma_{k_2}^2 \\
&= n_{k_1}\left(\sigma_{k_2}^2 - \sigma_{k_1}^2\right)\left(n_{k_1}+1\right) + \sigma_{k_1}^2\left(n_{k_1} - n_{k_2}\right)\left(n_{k_1}+n_{k_2}+1\right) \\
D_\Delta &= \left(n_{k_1}+1\right) n_{k_2} n_{k_1}\left(n_{k_2}+1\right).
\end{aligned}$$

Puisque $D_\Delta > 0$, la sélection suivante

$$\begin{cases}
\text{si } N_\Delta < 0 & \text{alors tester } \mathbf{z}_{k_1} \\
\text{si } N_\Delta > 0 & \text{alors tester } \mathbf{z}_{k_2} \\
\text{si } N_\Delta = 0 & \text{alors tester indifféremment } \mathbf{z}_{k_1} \text{ ou } \mathbf{z}_{k_2}
\end{cases}$$

maximise μ_g à l'étape $l+1$. □

Le théorème 4.2 montre que, quelle que soit l'alternative testée, la valeur de μ_g augmente toujours. Le théorème 4.3 indique que l'évolution de μ_g, pour toute stratégie n'explorant jamais les alternatives et testant toujours la même variable aléatoire, est bornée supérieurement par b^∞ qui est plus petit que $\mu_{[K]}$. Le théorème 4.4 étudie, connaissant les moyennes et les variances des alternatives, les situations d'exploration permettant une augmentation optimale de μ_g.

Ces propriétés concernant l'évolution de μ_g vont maintenant être testées empiriquement sur quatre problèmes créés de manière synthétique (voir tableau 4.1) avec un horizon de $H = 50$ étapes. Par la suite, nous entendrons par stratégie « *oracle* », une stratégie qui connaît les caractéristiques des alternatives. Trois algorithmes sont étudiés : (i) un algorithme oracle d'exploitation gloutonne qui teste toujours la meilleure alternative $\left[\widetilde{k} = [K]\right]$, (ii) un

	prob 1		*prob 2*		*prob 3*		*prob 4*	
	μ	σ	μ	σ	μ	σ	μ	σ
$\mathbf{z}_1$	0	2	0	1	0	2	0	0.01
$\mathbf{z}_2$	1	2	1	2	1	0.01	1	2

TABLE 4.1 – Quatre problèmes créés de manière synthétique pour étudier l'évolution de μ_g sur un horizon de $H = 50$ tests et quand $K = 2$ alternatives sont disponibles. Trois algorithmes sont étudiés : (i) un algorithme oracle d'exploitation gloutonne, (ii) un algorithme d'exploration qui teste cycliquement chaque alternative et (iii) l'algorithme optiK2. Pour chacun des quatre problèmes, les colonnes μ et σ contiennent respectivement la moyenne et l'écart-type des variables aléatoires.

algorithme de pure exploration qui teste les deux alternatives périodiquement $\left[\widetilde{k} = (l \mod 2) + 1\right]$ et (iii) un algorithme d'exploration basé sur les observations du théorème 4.4 et dénoté optiK2

$$\left[\text{si } N_\Delta < 0 \text{ alors } \widetilde{k} = 1 \text{ sinon } \widetilde{k} = 2\right].$$

Le processus expérimental se déroule comme suit. Initialement, les valeurs de n_1 et de n_2 sont égales à un. A chaque round l et selon l'une des trois stratégies, la valeur de $n_{\widetilde{k}}$ est incrémentée de un. La quantité μ_g est ensuite recalculée sur base de $\{\mu_k\}_{k\in[1,K]}$, de $\{\sigma_k\}_{k\in[1,K]}$ et de $\{n_k\}_{k\in[1,K]}$ (voir section 4.1). La figure 4.2 reprend l'évolution de μ_g pour les quatre problèmes synthétiques.

On observe que, dans tous les cas, la courbe de μ_g est non-décroissante (voir théorème 4.2). Cette observation ne sera plus systématiquement vraie lorsque plus de deux alternatives seront en compétition ($K > 2$). Voici quelques remarques sur l'évolution des courbes des trois algorithmes.

Commençons par la stratégie optiK2 qui se révèle, dans chacun des quatre problèmes, la meilleure stratégie (voir théorème 4.4). Nous observons dans le problème numéro un que les écarts-types des deux variables aléatoires sont égaux (voir tableau 4.1). Le premier terme de N_Δ s'annule et donc $N_\Delta = \sigma_1^2(n_1-n_2)(n_1+n_2+1)$ (voir équation (4.7) à la page 80). Comme σ_1^2 et $(n_1 + n_2 + 1)$ sont par définition positifs et comme la stratégie optiK2 choisit l'alternative à tester en fonction du signe de N_Δ, si $(n_1-n_2) > 0$ alors optiK2 teste $\mathbf{z}_2$ sinon c'est $\mathbf{z}_1$ qui est testé. L'algorithme optiK2 devient ici équivalent à l'algorithme de pure exploration qui teste les deux alternatives périodiquement. C'est la raison pour laquelle les courbes d'évolution de μ_g de l'algorithme optiK2 et de l'algorithme d'exploration se confondent dans le premier problème (voir figure 4.2). Dans le problème numéro quatre, l'écart-type de l'alternative non optimale est très faible (voir tableau 4.1). Le terme $n_1(\sigma_2^2 - \sigma_1^2)(n_1 + 1)$ dans N_Δ (voir équation (4.7)) est alors très grand et,

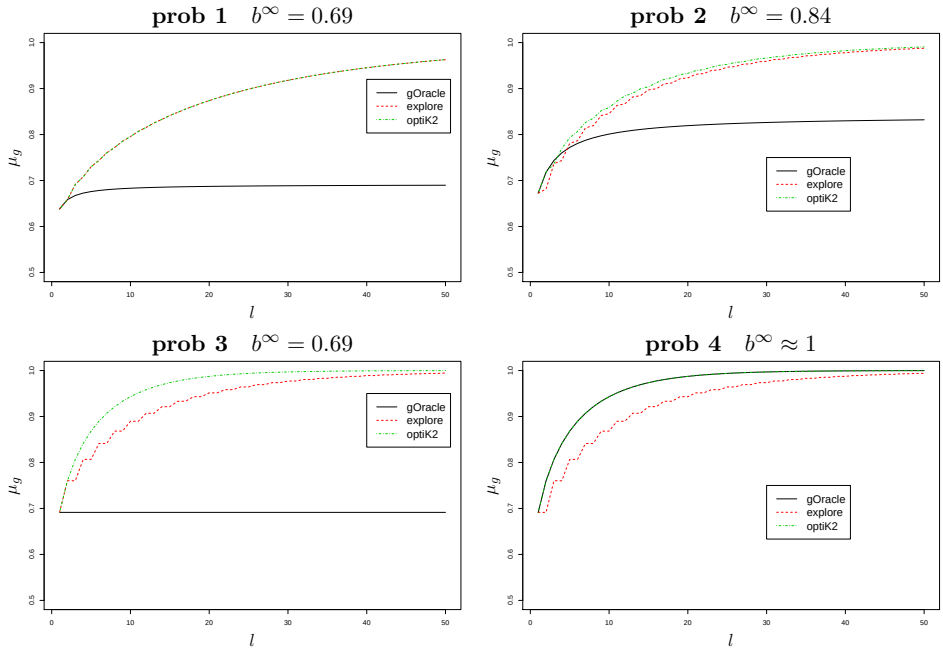


FIGURE 4.2 – Evolution de la quantité μ_g pour les quatre problèmes synthétiques sous une stratégie oracle d'exploitation gloutonne, une stratégie d'exploration et la stratégie optiK2. Pour chaque cas, la valeur de la borne supérieure b^∞ est donnée lorsque seul $\mathbf{z}_2$ est testé.

comme N_Δ reste positif, l'algorithme optiK2 devient donc équivalent à la stratégie oracle gloutonne qui teste toujours l'alternative $\mathbf{z}_2$ (voir figure 4.2).

Examinons maintenant l'évolution des courbes de μ_g pour la stratégie oracle gloutonne. Dans les problèmes un et deux, la valeur de μ_g augmente d'abord très légèrement pour ensuite se stabiliser à la valeur de b^∞. Dans le problème trois, cette augmentation n'est même pas perceptible. Les stratégies qui ne testent qu'une alternative voient donc effectivement leur évolution de μ_g bloquée par une borne supérieure pouvant être bien plus petite que l'optimum $\mu_{[K]}$.

Pour terminer, étudions les courbes d'évolution de μ_g sous une stratégie d'exploration testant les deux alternatives périodiquement. Dans ce cas, lorsque l tend vers l'infini, $n_{[K]}$ et $n_{[K-1]}$ tendent également tous les deux vers l'infini et nous avons

$$\lim_{l\to\infty} \sqrt{\frac{\sigma^2_{[K]}}{n_{[K]}} + \frac{\sigma^2_{[K-1]}}{n_{[K-1]}}} = 0$$

et donc (voir figure 4.1 à la page 75)

$$\lim_{l\to\infty} \Pr\{S_{[K]}\} = 1.$$

	cas 1		*cas 2*		*cas 3*		*cas 4*		*cas 5*		*cas 6*	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
$\mathbf{z}_1$	0	1	0	3	0	3	0	3	0	6	0	10
$\mathbf{z}_2$	0	1	0	1	0	3	0	0.1	0.2	3	0	10
$\mathbf{z}_3$	1	0.1	1	0.1	1	0.1	1	3	0.7	0.1	0.9	10
$\mathbf{z}_4$									1	3	1	10

TABLE 4.2 – Six problèmes synthétiques pour estimer l'évolution de μ_g sous une stratégie d'exploration où $K > 2$. Pour chaque cas, les colonnes μ et σ contiennent respectivement les moyennes et les écarts-types des alternatives.

Cela signifie qu'appliquer une multitude de fois des actions de pure exploration garantit que μ_g augmentera toujours et qu'asymptotiquement μ_g atteindra l'optimum $\mu_{[K]}$.

Dans le cas d'une stratégie testant toujours la même alternative, telle la stratégie oracle gloutonne, il faut donc explorer l'autre alternative afin de ne plus être bloqué par b^{∞}.

4.2.2 L'évolution de μ_g lorsque $K > 2$

Considérons maintenant le cas où l'ensemble des alternatives disponibles contient plus de deux éléments ($K > 2$). La quantité $\mu_g^{l+1}(k)$ est définie comme le prochain gain moyen d'une action gloutonne lorsque l'alternative k est testée au round l. Lorsque $K > 2$, nous allons montrer empiriquement que l'évolution de la quantité μ_g est plus complexe et que la propriété monotone $\mu_g^{l+1}(k) > \mu_g$ obtenue dans le théorème 4.2 pour le cas $K = 2$ n'est plus systématiquement observée. Nous allons rencontrer des cas où tester une alternative fera diminuer la valeur du gain espéré d'une action gloutonne à l'étape suivante. Afin de mettre en évidence ce type de comportement contre intuitif, l'évolution de μ_g sera étudiée sous une stratégie d'exploitation oracle gloutonne qui teste toujours la meilleure alternative $\left[\widetilde{k} = [K]\right]$ et sous une stratégie de pure exploration qui teste toutes les alternatives cycliquement $\left[\widetilde{k} = (l \mod K) + 1\right]$.

Considérons d'abord l'évolution de μ_g lorsqu'une stratégie d'exploration est utilisée. Six problèmes (voir tableau 4.2 et figure 4.3) créés de manière synthétique sont testés avec un horizon $H = 200$. Nous observons dans les six cas que la valeur de μ_g augmente toujours. Cette augmentation est aussi dans chaque cas plus forte au départ. Dans les cas cinq et six, nous observons une oscillation de μ_g en début de test. Cette oscillation sera expliquée plus loin dans le présent chapitre.

Considérons à présent l'évolution de μ_g lorsqu'une stratégie oracle gloutonne est utilisée. Nous allons montrer expérimentalement que la valeur de $\mu_g^{l+1}([K]) - \mu_g$ peut être négative dans certains cas. Ceci signifie que tester

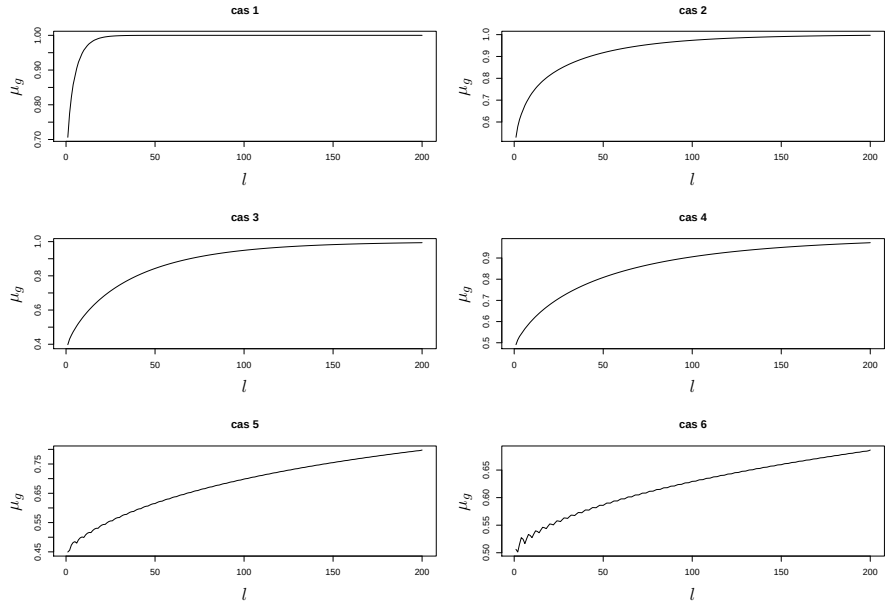


FIGURE 4.3 – Evolution de μ_g sous une stratégie d'exploration pour les six problèmes synthétiques définis dans le tableau 4.2.

la meilleure alternative peut réduire le gain espéré des prochaines actions gloutonnes. Nous verrons que cela se produit lorsque la quantité $\frac{\sigma}{\sqrt{n}}$ des mauvaises alternatives est grande et où $\frac{\sigma_k}{\sqrt{n_k}}$ est l'écart-type de l'estimateur de moyenne $(\widehat{\boldsymbol{\mu}}_k)$.

Six cas créés de manière synthétique sont testés (voir tableau 4.3 et figure 4.4). Les différents cas se distinguent par (i) le nombre d'alternatives disponibles, (ii) les moyennes des différentes alternatives et (iii) les valeurs $\frac{\sigma}{\sqrt{n}}$ des alternatives sous-optimales. Dans ces six cas, le paramètre $n_{[K]}$ – qui contient le nombre de tests réalisés sur la meilleure alternative – prend ses valeurs dans l'intervalle $[1, 200]$ et l'écart-type $\sigma_{[K]}$ de cette alternative est toujours fixé à dix [2].

Dans les problèmes un et cinq, les écarts-types $\frac{\sigma}{\sqrt{n}}$ des moyennes arithmétiques des mauvaises alternatives possèdent une faible valeur (voir tableau 4.3). La courbe de μ_g montre (voir figure 4.4) que tester l'alternative optimale (c'est-à-dire augmenter la valeur de $n_{[K]}$) améliore la valeur de la quantité μ_g. Nous constatons l'opposé dans les problèmes deux et six. Si les valeurs $\frac{\sigma}{\sqrt{n}}$ des alternatives les moins compétitives sont grandes, alors tester l'alternative optimale $\mathbf{z}_{[K]}$ diminue la valeur de μ_g.

Les cas trois et quatre sont des situations intermédiaires où l'évolution de

2. Cela signifie donc également que $\frac{\sigma_{[K]}}{\sqrt{n_{[K]}}}$ prend ses valeurs dans l'intervalle $[10, 10/\sqrt{200}]$.

	cas 1		*cas 2*		*cas 3*		*cas 4*		*cas 5*		*cas 6*	
	μ	$\frac{\sigma}{\sqrt{n}}$	μ	$\frac{\sigma}{\sqrt{n}}$	μ	$\frac{\sigma}{\sqrt{n}}$	μ	$\frac{\sigma}{\sqrt{n}}$	μ	$\frac{\sigma}{\sqrt{n}}$	μ	$\frac{\sigma}{\sqrt{n}}$
$\mathbf{z}_1$	0	0.0001	0	5	0	1	0	1	0	0.0001	0	5
$\mathbf{z}_2$	0	0.0001	0	5	0	10	0	1	0.2	0.0001	0.2	5
$\mathbf{z}_3$	1	$*$	1	$*$	1	$*$	0	10	0.7	0.0001	0.7	10
$\mathbf{z}_4$							1	$*$	1	$*$	1	$*$

TABLE 4.3 – Six problèmes synthétiques pour estimer l'évolution de μ_g sous un algorithme oracle d'exploitation gloutonne où $K > 2$. Pour chaque cas, la première colonne contient la moyenne des variables aléatoires et la deuxième colonne contient l'écart-type des estimateurs des moyennes de ces variables. Un astérisque ($*$) dans l'un des champs du tableau signifie que, durant les expériences, cette valeur évolue entre 10 et $10/\sqrt{200}$.

μ_g est plus complexe. Au début (c'est-à-dire lorsque $\frac{\sigma_{[K]}}{\sqrt{n_{[K]}}}$ est grand), quand on teste l'alternative optimale, la valeur de μ_g commence par diminuer. Ensuite, lorsque $\frac{\sigma_{[K]}}{\sqrt{n_{[K]}}}$ devient suffisamment petit, tester l'optimum fait à nouveau augmenter μ_g. Nous constatons, comme pour le problème quatre, que la remontée de μ_g peut être très lente.

A l'exception des problèmes un et cinq, l'évolution de μ_g, lorsque $[K]$ est testé, a toujours montré une décroissance à l'un ou l'autre moment.

Etudions maintenant le cas six plus en détails (voir tableau 4.3). $\mathbf{z}_1$ et $\mathbf{z}_2$ sont les deux alternatives les moins performantes. Pour ces deux variables, l'écart-type de la moyenne arithmétique est fixé à cinq. Nous supposons ici que $\sigma_1 = \sigma_2 = 5$ et $n_1 = n_2 = 1$. La variable $\mathbf{z}_3$, avec une espérance de 0.7, constitue la seconde meilleure variable. La quantité $\sigma_3/\sqrt{n_3}$ vaut dix et nous supposons que $\sigma_3 = 10$ et $n_3 = 1$. La dernière alternative $\mathbf{z}_4$ est l'optimum. Son espérance est fixée à un et son écart-type équivaut à dix. Dans la figure 4.4, n_4 prend ses valeurs dans l'intervalle $[1, 200]$ et nous observons une décroissance de μ_g lorsque n_4 augmente.

Etudions maintenant l'évolution de μ_g dans le cas six mais lorsque les deux meilleures variables sont testées et jamais les deux autres. La figure 4.5 donne la valeur de μ_g en fonction de $n_{[K]}$ et $n_{[K-1]}$ lorsque ces deux quantités prennent leurs valeurs dans l'intervalle $[1, 200]$. Nous observons que μ_g continue de décroître quelle que soit la valeur donnée au couple $(n_{[K]}, n_{[K-1]})$. Il est alors impossible de faire augmenter μ_g en testant ces deux alternatives optimales.

Considérons maintenant une situation identique à l'exception que cette fois deux-cents tests sont réalisés sur les deux variables sous-optimales ($\mathbf{z}_1$ et $\mathbf{z}_2$). L'écart-type de leurs deux moyennes arithmétiques ($\widehat{\boldsymbol{\mu}}_1$ et $\widehat{\boldsymbol{\mu}}_2$) devient donc $5/\sqrt{200} \approx 0.35$. La figure 4.6 présente à nouveau l'évolution de la quantité μ_g lorsque $n_{[K]}$ et $n_{[K-1]}$ prennent leurs valeurs dans $[1, 200]$. Ici par contre, μ_g croît lorsque les deux alternatives optimales sont testées.

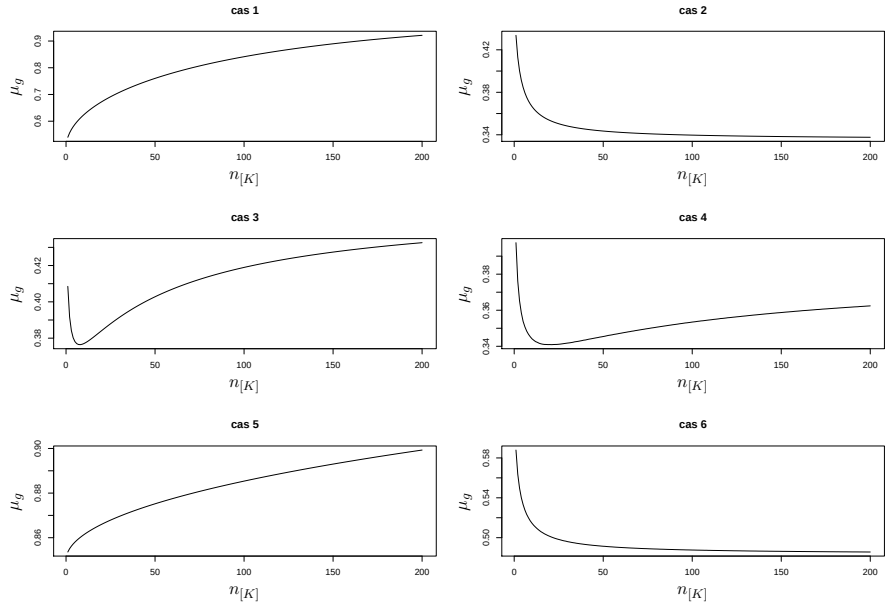


FIGURE 4.4 – Evolution de μ_g sous une stratégie oracle gloutonne pour les six problèmes synthétiques définis dans le tableau 4.3. Pour les quatre premiers problèmes, nous avons $\mu_g = \Pr\{S_{[K]}\}$.

Cette étude du cas numéro six à montré qu'avant de tester les alternatives optimales, il est nécessaire de se placer dans une condition favorable où les alternatives sous-optimales sont testées suffisamment de fois pour que l'écart-type de leur moyenne arithmétique soit relativement faible.

L'oscillation observée dans les cas cinq et six de la figure 4.3, lorsqu'un algorithme d'exploration est appliqué, peut maintenant être expliquée. Cet algorithme sélectionne chaque alternative de manière périodique. Les alternatives optimales sont donc régulièrement testées. Dans la phase initiale de ces deux cas, les quantités $\sigma/\sqrt{n}$ des mauvaises alternatives sont élevées. La valeur de μ_g est alors réduite lorsqu'une alternative optimale est testée. Il est à noter que cette oscillation se réduit au cours du temps puisque les quantités $\sigma/\sqrt{n}$ des alternatives sous-optimales diminuent.

4.2.2.1 L'évolution de $\Pr\{S_{[K]}\}$ lorsque $K > 2$

Afin d'expliquer la diminution de μ_g lorsque $[K]$ est testé, étudions maintenant cette évolution du point de vue de la probabilité d'effectuer une sélection correcte $\Pr\{S_{[K]}\}$. Nous remarquons dans les quatre premiers problèmes que la moyenne de l'alternative optimale $\mu_{[K]}$ est de un et que les autres alternatives présentent une moyenne de zéro (voir tableau 4.3). Comme $\mu_g = \sum \Pr\{S_k\}\,\mu_k$, dans ces quatre problèmes le gain moyen d'une action gloutonne μ_g est ici égal à la probabilité de faire une sélection correcte

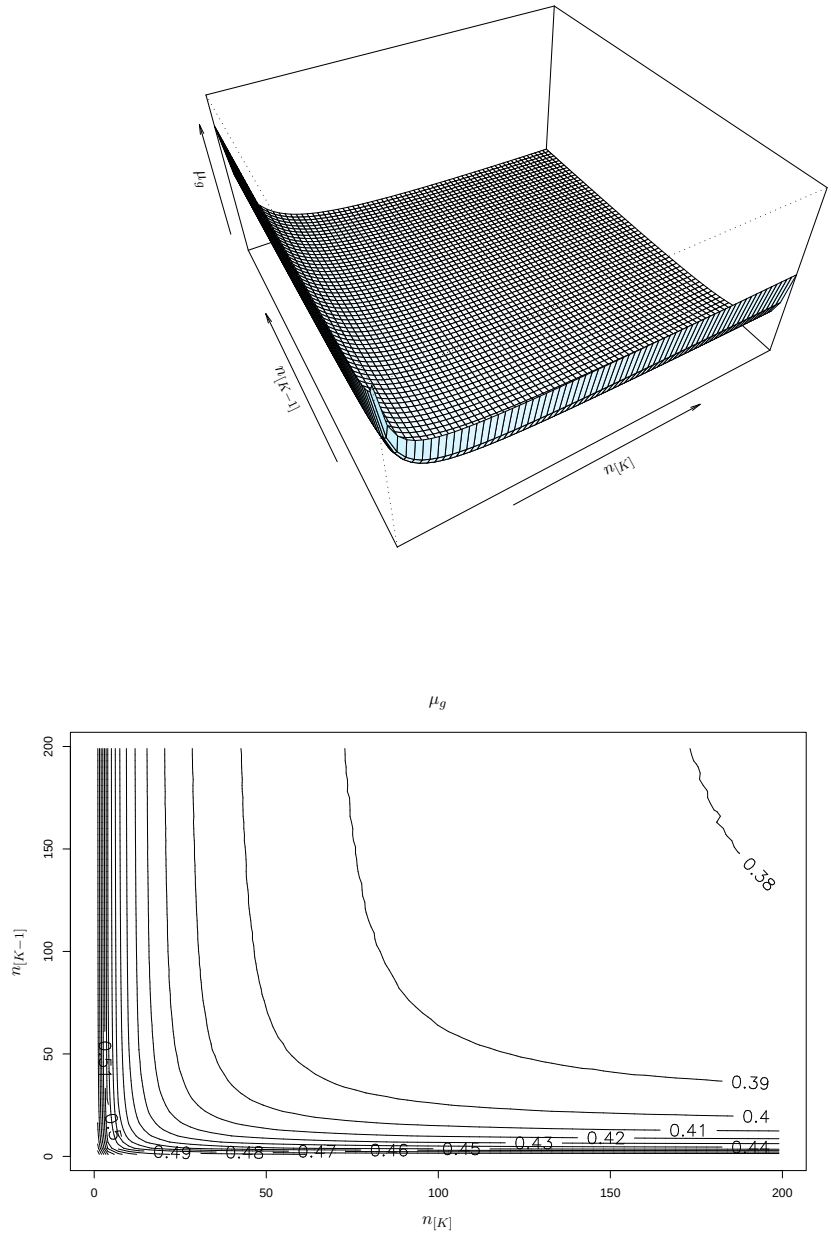


FIGURE 4.5 – Evolution de la quantité μ_g lorsque les deux alternatives $[K]$ et $[K-1]$ sont testées. Ici, les quantités $\sigma/\sqrt{n}$ des alternatives sous-optimales sont élevées, ce qui fait décroître μ_g.

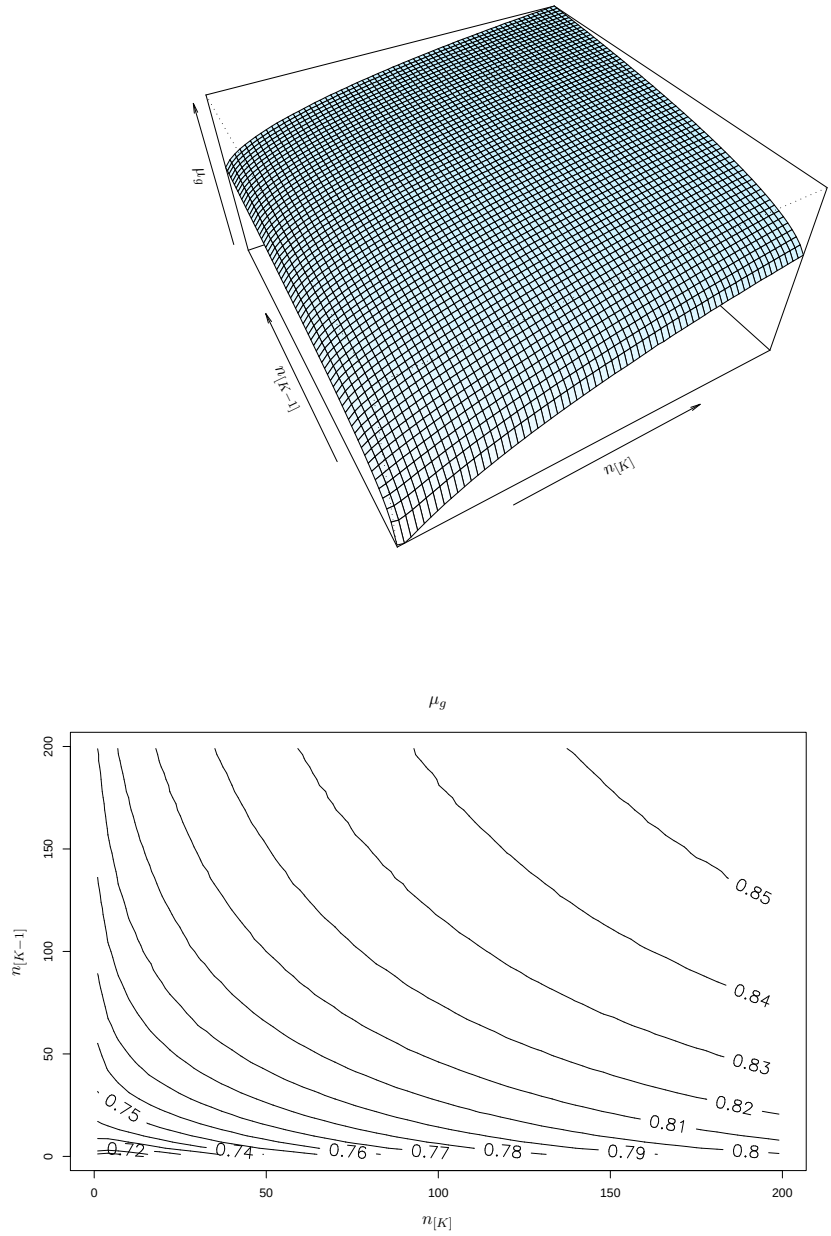


FIGURE 4.6 – Evolution de la quantité μ_g lorsque les deux alternatives $[K]$ et $[K-1]$ sont testées. Ici, les quantités $\sigma/\sqrt{n}$ des alternatives sous-optimales sont faibles, ce qui fait croître μ_g.

$\Pr\{S_{[K]}\}$.

Dans le premier problème, sous une stratégie oracle gloutonne, l'évolution de $\Pr\{S_{[K]}\}$ est une courbe monotone croissante. Dans le deuxième problème, l'évolution de $\Pr\{S_{[K]}\}$ suit une courbe monotone décroissante. Enfin dans les problèmes trois et quatre, l'évolution de $\Pr\{S_{[K]}\}$ s'apparente à une courbe plus complexe qui décroît pour ensuite remonter.

Comme nous l'avons déjà vu, la probabilité d'opérer un choix correct via une sélection gloutonne vaut

$$\begin{aligned}
&\Pr\{S_{[K]}\} \\
&= \Pr\Big\{ \widehat{\boldsymbol{\mu}}_{[K]} > \widehat{\boldsymbol{\mu}}_1 \ , \ \ldots \ , \ \widehat{\boldsymbol{\mu}}_{[K]} > \widehat{\boldsymbol{\mu}}_{[K]-1} \ , \ \widehat{\boldsymbol{\mu}}_{[K]} > \widehat{\boldsymbol{\mu}}_{[K]+1} \ , \ \ldots \ , \ \widehat{\boldsymbol{\mu}}_{[K]} > \widehat{\boldsymbol{\mu}}_K \Big\} \qquad (4.8) \\
&= \Pr\Big\{ \widehat{\boldsymbol{\mu}}_{[K]} > \mathbf{X}_{max} \Big\} \qquad (4.9)
\end{aligned}$$

où

$$\mathbf{X}_{max} = \max_{k \in \{1K\}/[K]} \widehat{\boldsymbol{\mu}}_k.$$

Selon l'équation (4.8), la probabilité d'effectuer une sélection correcte est égale à la probabilité que $\widehat{\boldsymbol{\mu}}_{[K]}$ ait la plus grande valeur au sein de l'ensemble des K estimations $\{\widehat{\boldsymbol{\mu}}_1, \ldots, \widehat{\boldsymbol{\mu}}_K\}$.

Dans l'équation (4.9), le problème est reformulé comme suit. La probabilité d'effectuer une sélection correcte devient la probabilité que $\widehat{\boldsymbol{\mu}}_{[K]}$ ait la plus grande valeur dans un ensemble de seulement deux variables $\Big\{\widehat{\boldsymbol{\mu}}_{[K]}, \mathbf{X}_{max}\Big\}$ où $\mathbf{X}_{max}$ est le maximum de l'ensemble $\{\widehat{\boldsymbol{\mu}}_k\}$ avec $k \in \{1, \ldots, K\}/[K]$ (voir annexe G). La figure 4.7 modélise la distribution de probabilités de la variable $\mathbf{X}_{max}$ pour les quatre problèmes.

Il est à noter que l'espérance de $\mathbf{X}_{max}$ dépend des écarts-types des moyennes arithmétiques des mauvaises alternatives ($\sigma/\sqrt{n}$). Dans le problème un, les écarts-types de $\widehat{\boldsymbol{\mu}}_1$ et $\widehat{\boldsymbol{\mu}}_2$ sont petits et $\mathbb{E}[\mathbf{X}_{max}]$ est par conséquent également peu élevé. Dans le problème deux, les grandes variances de $\widehat{\boldsymbol{\mu}}_1$ et $\widehat{\boldsymbol{\mu}}_2$ engendrent une espérance de $\mathbf{X}_{max}$ élevée.

Nous allons fournir une explication sur l'évolution de la quantité $\Pr\{S_{[K]}\}$ dans les deux premiers problèmes, c'est-à-dire lorsque les $\frac{\sigma}{\sqrt{n}}$ des mauvaises alternatives sont égales entre elles.

Dans le premier problème, l'alternative optimale est donc confrontée à une alternative $\mathbf{X}_{max}$ ayant une petite moyenne $\big(\mathbb{E}[\mathbf{X}_{max}] \approx 6 \times 10^{-5}\big)$. La seule possibilité pour l'alternative $\mathbf{X}_{max}$ de « *battre* » $\widehat{\boldsymbol{\mu}}_{[K]}$ est que l'observation de $\widehat{\boldsymbol{\mu}}_{[K]}$ soit plus petite que celle de $\mathbf{X}_{max}$. Tester l'alternative $[K]$ réduit la variance de $\widehat{\boldsymbol{\mu}}_{[K]}$ et comme $\mathbb{E}\Big[\widehat{\boldsymbol{\mu}}_{[K]}\Big] > \mathbb{E}[\mathbf{X}_{max}]$, tester $[K]$ augmente aussi la probabilité de faire une sélection correcte $\Pr\{S_{[K]}\}$.

Dans le deuxième problème, l'alternative $\mathbf{X}_{max}$ possède une plus grande moyenne que l'alternative optimale $\Big(1 = \mathbb{E}\Big[\widehat{\boldsymbol{\mu}}_{[K]}\Big] < \mathbb{E}[\mathbf{X}_{max}] \approx 2.82\Big)$. Une

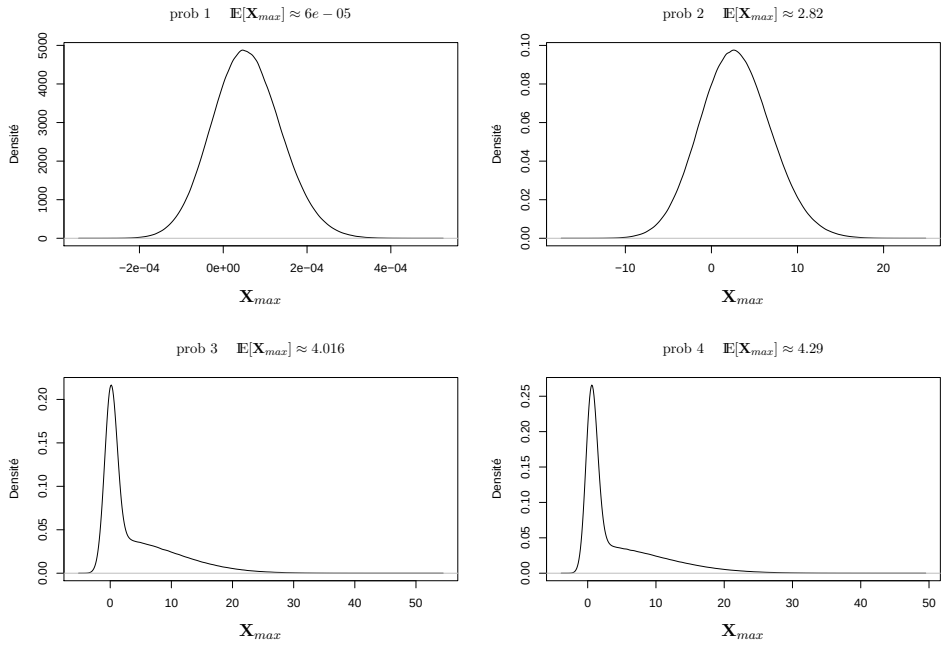


FIGURE 4.7 – Estimation par Monte-Carlo des distributions de probabilités de $\mathbf{X}_{max}$ pour les quatre premiers problèmes définis dans le tableau 4.3.

réalisation de la moyenne arithmétique $\widehat{\boldsymbol{\mu}}_{[K]}$ a plus de chances d'être supérieure à une réalisation de $\mathbf{X}_{max}$ si la variance de $\widehat{\boldsymbol{\mu}}_{[K]}$ est grande. Tester l'alternative $\mathbf{z}_{[K]}$ réduit la variance de $\widehat{\boldsymbol{\mu}}_{[K]}$ et diminue donc également la probabilité $\Pr\{S_{[K]}\}$ de faire une sélection correcte.

Pour les problèmes trois et quatre, la densité de $\mathbf{X}_{max}$ est davantage irrégulière. Ceci explique l'évolution plus complexe de μ_g qui commence par décroître pour ensuite remonter.

4.3 Calcul de la probabilité de sélectionner l'alternative k par Monte Carlo

La section 4.1 a donné une définition analytique de la probabilité $\Pr\{S_k\}$. Celle-ci demande de calculer la probabilité d'une distribution normale multivariée. Cette section va d'abord présenter l'approche numérique proposée par Genz qui permet le calcul d'une telle distribution normale multivariée. Une implémentation basique d'une approche Monte Carlo [3] sera ensuite proposée et les deux approches seront testées sur un problème synthétique de calcul de $\Pr\{S_k\}$. La section se termine par quelques considérations concer-

3. Nous utilisons le terme *basique* pour signifier qu'aucune optimisation n'a été réalisée afin d'améliorer l'exactitude des estimations renvoyées par la méthode Monte Carlo.

nant le calcul de $\Pr\{S_k^-\}$ dans les problèmes d'apprentissage artificiel.

Pour obtenir $\Pr\{S_k\}$, il faut calculer une distribution d'une normale multivariée. Dans cette thèse, nous allons utiliser la méthode de Genz [40, 41] qui a été implémentée dans la fonction *pmvnorm* de la librairie *mvtnorm* [4] du langage de programmation R [5].

Soit $[\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_K]^T$, un vecteur de K éléments distribués selon une loi normale multivariée de moyenne Γ et de matrice de covariance Σ, et soit la probabilité [83]

$$\begin{aligned} &\Pr\{a_1 < \mathbf{s}_1 < b_1 \ , \ \ldots \ , \ a_K < \mathbf{s}_K < b_K\} \\ &= \frac{1}{\sqrt{|\Sigma| \, (2\pi)^2}} \int_{a_1}^{b_1} \int_{a_2}^{b_2} \cdots \int_{a_K}^{b_K} e^{-\frac{1}{2}(\bar{x}-\Gamma)^T \Sigma^{-1} (\bar{x}-\Gamma)} \, d\bar{x}. \end{aligned} \tag{4.10}$$

où $\bar{x} = [x_1, x_2, \ldots x_K]$ et $|\Sigma|$ désigne le déterminant de la matrice Σ.

L'algorithme de Genz permet de calculer ce type d'intégrale multiple. L'algorithme est divisé en deux phases. Une méthode est d'abord appliquée sur (4.10) pour transformer l'intégration initiale en une intégration dans un hypercube unitaire [40]. Dans cette situation, différentes méthodes d'intégration numérique peuvent alors être appliquées. Pour calculer cette intégrale, la fonction *pmvnorm* utilise une approche *quasi-Monte Carlo* [41] qui échantillonne l'espace d'intégration d'une manière plus uniforme que les approches purement aléatoires.

Le nombre de cycles de l'approche *quasi-Monte Carlo* de la fonction *pmvnorm* dépend de R_{Genz} et de ϵ_{Genz} où la quantité R_{Genz} est le nombre de cycles maximal et la quantité ϵ_{Genz} est l'erreur maximale tolérée. Si ϵ_{Genz} est nul alors la complexité de l'algorithme proposé par Genz est en $O(K \times R_{Genz})$ [41].

Il est également possible de calculer la valeur de la probabilité de sélection $\Pr\{S_k\}$ en simulant le problème au moyen d'une implémentation basique de la méthode Monte Carlo (voir l'algorithme 11). Nous savons que la moyenne arithmétique $\widehat{\boldsymbol{\mu}}_k$ de n_k observations suit une distribution normale de moyenne μ_k et d'écart-type $\sigma_k/\sqrt{n_k}$. Pour chaque alternative, une observation de $\widehat{\boldsymbol{\mu}}_k$ est collectée (ligne 5) et $[\widehat{\mathbf{K}}]$ constitue l'alternative ayant la plus grande valeur de $\widehat{\boldsymbol{\mu}}$ (ligne 7). Cette procédure est répétée R fois (lignes 3 à 9) et l'on conserve dans le vecteur **compteur** le nombre d'occurrences optimales de chaque alternative (ligne 8). La probabilité de sélection s'obtient en divisant le nombre de fois que chacune des alternatives s'est montrée optimale par le nombre de répétitions R (ligne 10). Cette méthode donne donc un vecteur de taille K qui contient – pour chaque alternative – une estimation de la probabilité qu'elle présente la moyenne arithmétique la plus élevée.

4. `http://cran.r-project.org/web/packages/mvtnorm/index.html`
5. `http://www.r-project.org/`

Algorithme 11 Calcul de $\Pr\{S_k\}$ par une méthode Monte Carlo basique

1: Fixer la valeur de R.
2: $\forall k \in [1, \ldots, K]$, $\mathbf{compteur}_k \leftarrow 0$.
3: **pour** $r = 1$ à R **faire**
4: **pour** $k = 1$ à K **faire**
5: Prendre une observation $\widehat{\boldsymbol{\mu}}_k$ où $\widehat{\boldsymbol{\mu}}_k \sim N\left(\ \mu_k\ ,\ \sigma_k/\sqrt{n_k}\ \right)$.
6: **fin pour**
7: $[\widehat{\mathbf{K}}] = \arg\max \widehat{\boldsymbol{\mu}}_k$.
8: $\mathbf{compteur}_{[\widehat{K}]} \leftarrow \mathbf{compteur}_{[\widehat{K}]} + 1$.
9: **fin pour**
10: $\mathbf{Pr}\{S_k\} \leftarrow \mathbf{compteur}_k/R$.

Deux méthodes pour le calcul de $\Pr\{S_k\}$ ont donc été proposées : la méthode analytique décrite dans le théorème 4.1 qui utilise l'approche de Genz, et la méthode par Monte Carlo basique. Ces deux approches vont être comparées expérimentalement sur l'exemple synthétique de la section 4.1. Dans cet exemple, le vecteur Γ et la matrice Σ ont été calculés et valent :

$$\Gamma = [1, -1, 0]^T,$$

$$\Sigma = \begin{pmatrix} 3/5 & 2/5 & 2/5 \\ 2/5 & 1/2 & 2/5 \\ 2/5 & 2/5 & 7/10 \end{pmatrix}.$$

La fonction *pmvnorm* est utilisée pour calculer la probabilité que les trois variables $\widehat{\mathbf{r}}_1$, $\widehat{\mathbf{r}}_2$ et $\widehat{\mathbf{r}}_3$ soient toutes positives. Cette fonction *pmvnorm* permet de modifier la valeur de R_{Genz}. Nous avons testé les performances de cette fonction où R_{Genz} prend les vingt-cinq valeurs suivantes : $100, 2100, 4100, \ldots, 48100$ [6]. Pour chacune des valeurs de R_{Genz}, l'algorithme a été utilisé deux-cents fois. Cette série de tests permet de calculer la variance des valeurs renvoyées par *pmvnorm* lorsque R_{Genz} évolue (voir partie gauche de la figure 4.8).

Le paramètre R de l'algorithme 11 définit le nombre de cycles de l'approche Monte Carlo. Nous avons testé les performances de cette méthode où R prend les vingt-cinq mêmes valeurs : $100, 2100, 4100, \ldots, 48100$. Pour chacune des valeurs de R, l'algorithme a également été utilisé deux-cents fois et l'évolution de la variance des résultats se trouve dans la partie droite de la figure 4.8.

En utilisant le fait que, pour chaque valeur de R_{Genz} et de R, deux-cents tests ont été réalisés, nous avons observé – via un F-test d'égalité de variance – que les valeurs renvoyées par la méthode analytique ont une variance significativement inférieure aux valeurs obtenues par l'autre méthode.

6. Le paramètre ϵ_{Genz} a été fixé à zéro.

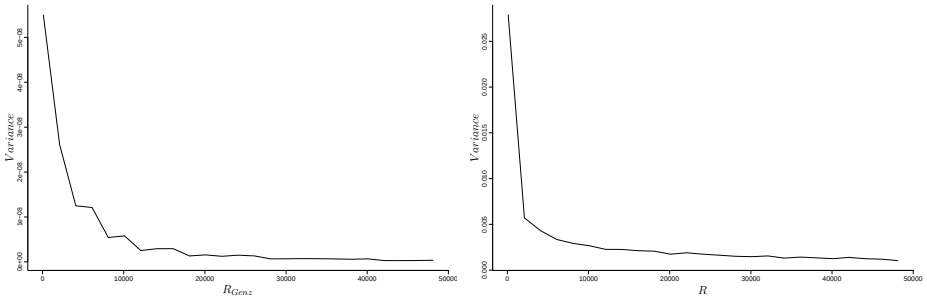


FIGURE 4.8 – Evolution de la variance du calcul de $\Pr\{S_2\}$ lorsque le nombre de cycles R_{Genz} et R évolue. La partie de gauche contient les résultats lorsque c'est la méthode analytique décrite dans le théorème 4.1 qui est utilisée et la partie de droite contient les résultats lorsque c'est la méthode par Monte Carlo basique qui est utilisée.

Dans le cas de l'apprentissage artificiel, aucune définition analytique de $\Pr\{S_k^-\}$ n'a pu – à ce jour – être établie. Cette quantité peut toutefois être calculée via la même technique de Monte Carlo basique. Les étapes en sont décrites dans l'algorithme 12. Pour utiliser cette technique, il faut toutefois connaître les valeurs de $\mu_v = \mathbb{E}\Big[\Big(\mathbf{y}_i - h(\mathbf{x}_i, \boldsymbol{\alpha}^k_{N(i)})\Big) \mid \Lambda_k\Big]$ et de $\sigma_v^2 = \mathrm{Var}\Big[\Big(\mathbf{y}_i - h(\mathbf{x}_i, \boldsymbol{\alpha}^k_{N(i)})\Big) \mid \Lambda_k\Big]$ qui sont en règle générale non disponibles.

4.4 Estimation du gain espéré d'une action gloutonne

Les sections précédentes ont proposé une définition analytique de μ_g, une quantité importante pour résoudre les problèmes de sélection dans un environnement aléatoire. Nous avons supposé que les moyennes et les écarts-types des variables étaient connus mais cette hypothèse s'avère irréaliste puisque dans les problèmes de sélection étudiés dans cette thèse, ces deux valeurs ne sont jamais accessibles. Cette supposition sera donc ici abandonnée au profit de techniques d'estimation. Il est à noter que contrairement aux algorithmes décrits dans la section 3.2, nous n'utilisons pas de bornes sur la quantité $\Pr\{S_{[K]}\}$ mais nous cherchons à estimer μ_g. Dans la présente section, cinq estimateurs du gain moyen espéré d'une action gloutonne μ_g construits sur les observations $\{\mathbf{Z}_k(l)\}_{k=1,\dots,K}$ vont être proposés et seront ensuite comparés expérimentalement au point 4.4.5.

Nous avons vu – à la section 4.1 – que la quantité μ_g est une fonction de $\{\mu_k\}$, $\{\sigma_k\}$ et $\{n_k\}$ avec $k \in [1, \dots, K]$. Comme μ_g est estimé sur base des observations $\{\mathbf{Z}_k(l)\}_{k=1,\dots,K}$, seules les quantités $\{\mu_k\}_{k=1,\dots,K}$ et $\{\sigma_k\}_{k=1,\dots,K}$

Algorithme 12 Calcul de $\Pr\{S_k^-\}$ par Monte Carlo

1: Fixer la valeur de R.
2: $\forall k \in [1, \ldots, K]$, $\mathbf{compteur}_k \leftarrow 0$.
3: **pour** $r = 1$ à R **faire**
4: **pour** $k = 1$ à K **faire**
5: Prendre une observation $\widehat{\mathbf{MISE}}_{loo}(\Lambda_k)$ où

$$\widehat{\mathbf{MISE}}_{loo}(\Lambda_k) = \sum_{i=1}^{N} (\sigma_v \cdot \mathbf{p} + \mu_v)^2$$

et où $\mathbf{p} \sim N(0, 1)$.
6: **fin pour**
7: $\mathbf{m} = \arg\min \widehat{\boldsymbol{\mu}}_k$.
8: $\mathbf{compteur}_{\mathbf{m}} \leftarrow \mathbf{compteur}_{\mathbf{m}} + 1$.
9: **fin pour**
10: $\mathbf{Pr}\{S_k^-\} \leftarrow \mathbf{compteur}_k / R$.

sont inconnues et $\{n_k\}$ est fixe.

4.4.1 L'estimateur *naïf*

La manière la plus simple de construire un estimateur de la quantité μ_g consiste à prendre les moyennes arithmétiques des observations collectées et à choisir $\widehat{\boldsymbol{\mu}}_{[\widehat{\mathbf{K}}]}$ comme estimation de μ_g, où $[\widehat{\mathbf{K}}]$ est l'alternative renvoyée par une action gloutonne $[\widehat{\mathbf{K}}] = \arg\max \widehat{\boldsymbol{\mu}}_k$. Malheureusement, il est facile de voir qu'un tel estimateur, dénoté $\widehat{\boldsymbol{\mu}}_g^{max}$, est biaisé.

Soit K variables aléatoires : $\mathbf{X}_1, \ldots, \mathbf{X}_K$; on peut montrer que [54]

$$\mathbb{E}[\max\{\mathbf{X}_1, \ldots, \mathbf{X}_K\}] > \max\{\mathbb{E}[\mathbf{X}_1], \ldots, \mathbb{E}[\mathbf{X}_K]\}$$

et, si nous remplaçons les variables $\mathbf{X}$ par $\widehat{\boldsymbol{\mu}}$, nous pouvons voir que

$$\mathbb{E}\big[\widehat{\boldsymbol{\mu}}_g^{max}\big] = \mathbb{E}[\max\{\widehat{\boldsymbol{\mu}}_1, \ldots, \widehat{\boldsymbol{\mu}}_K\}] > \max\{\mathbb{E}[\widehat{\boldsymbol{\mu}}_1], \ldots, \mathbb{E}[\widehat{\boldsymbol{\mu}}_K]\} = \mu_{[K]}.$$

Comme $\mu_{[K]} \geq \mu_g$, l'estimateur est biaisé : $\mathbb{E}\big[\widehat{\boldsymbol{\mu}}_g^{max}\big] > \mu_g$.

4.4.2 L'estimateur par *plug-in*

Une alternative à l'approche *naïve* présentée dans la section précédente consiste à utiliser les techniques de *plug-in* [33], c'est-à-dire à remplacer dans la définition analytique de μ_g (voir section 4.1) les termes $\{\mu_k\}$ et $\{\Pr\{S_k\}\}$ par leurs estimateurs $\{\widehat{\boldsymbol{\mu}}_k\}$ et $\{\widehat{\mathbf{Pr}}\{S_k\}\}$. La quantité $\widehat{\mathbf{Pr}}\{S_k\}$ est obtenue

en remplaçant dans (4.2) à la page 72 les termes μ et σ par $\widehat{\boldsymbol{\mu}}$ et $\widehat{\boldsymbol{\sigma}}$. Nous obtenons alors le nouvel estimateur comme suit

$$\widehat{\boldsymbol{\mu}}_g^{PL} = \sum_{k=1}^{K} \widehat{\mathbf{Pr}}\{S_k\} \cdot \widehat{\boldsymbol{\mu}}_k.$$

Le théorème 4.5 suivant montre que l'estimateur $\widehat{\boldsymbol{\mu}}_g^{PL}$ est également biaisé.

Théorème 4.5. *Soit un ensemble de K variables aléatoires $\{\widehat{\boldsymbol{\mu}}_1, \ldots, \widehat{\boldsymbol{\mu}}_K\}$. Si $\widehat{\boldsymbol{\mu}}_g^{PL} = \sum_{k=1}^{K} \widehat{\mathbf{Pr}}\{S_k\} \cdot \widehat{\boldsymbol{\mu}}_k$ est un estimateur par plug-in du gain espéré d'une action gloutonne μ_g alors $\mathbb{E}\left[\widehat{\boldsymbol{\mu}}_g^{PL}\right] > \mu_g$.*

Démonstration.

$$\begin{aligned}
\mathbb{E}\left[\widehat{\boldsymbol{\mu}}_g^{PL}\right] - \mu_g &= \mathbb{E}\left[\sum_{k=1}^{K} \widehat{\mathbf{Pr}}\{S_k\} \cdot \widehat{\boldsymbol{\mu}}_k\right] - \sum_{k=1}^{K} \Pr\{S_k\} \cdot \mu_k \\
&= \sum_{k=1}^{K} \mathbb{E}\left[\widehat{\mathbf{Pr}}\{S_k\} \cdot \widehat{\boldsymbol{\mu}}_k\right] - \sum_{k=1}^{K} \Pr\{S_k\} \cdot \mu_k \\
&= \sum_{k=1}^{K} \left(\mathbb{E}\left[\widehat{\mathbf{Pr}}\{S_k\} \cdot \widehat{\boldsymbol{\mu}}_k\right] - \Pr\{S_k\} \cdot \mu_k\right) \\
&= \sum_{k=1}^{K} \operatorname{cov}\left(\widehat{\mathbf{Pr}}\{S_k\}, \widehat{\boldsymbol{\mu}}_k\right)
\end{aligned}$$

où $\operatorname{cov}(\cdot,\cdot)$ est la fonction de covariance et $\widehat{\mathbf{Pr}}\{S_k\}$ est la probabilité estimée que k soit l'alternative possédant la plus grande moyenne arithmétique. Comme $\widehat{\mathbf{Pr}}\{S_k\}$ est elle-même fonction de la quantité $\widehat{\boldsymbol{\mu}}_k$ et que $\widehat{\mathbf{Pr}}\{S_k\}$ augmente avec cette quantité, les estimateurs $\widehat{\mathbf{Pr}}\{S_k\}$ et $\widehat{\boldsymbol{\mu}}_k$ sont corrélés positivement et, par conséquence, le terme de biais est positif. □

4.4.3 L'estimateur par *mise de côté*

Comme démontré dans le théorème 4.5, le biais de l'estimateur $\widehat{\boldsymbol{\mu}}_g^{PL}$ est dû à la corrélation entre les deux estimateurs $\widehat{\mathbf{Pr}}\{S_k\}$ et $\widehat{\boldsymbol{\mu}}_k$. Cette section propose une technique d'estimation de μ_g qui supprime la corrélation entre les termes $\widehat{\mathbf{Pr}}\{S_k\}$ et $\widehat{\boldsymbol{\mu}}_k$ en divisant l'ensemble des observations en deux parties distinctes. Nous définissons

$$\widehat{\boldsymbol{\mu}}_k^A = \frac{1}{\lfloor n_k/2 \rfloor} \sum_{j=1}^{\lfloor n_k/2 \rfloor} \mathbf{z}_k^j \tag{4.11}$$

$$\widehat{\boldsymbol{\sigma}}_k^A = \sqrt{\frac{1}{\lfloor n_k/2 \rfloor - 1} \sum_{j=1}^{\lfloor n_k/2 \rfloor} (\mathbf{z}_k^j - \widehat{\boldsymbol{\mu}}_k^A)^2} \tag{4.12}$$

comme étant respectivement les estimations des espérances et des écarts-types des variables aléatoires calculées sur la première moitié des observations, et nous définissons

$$\widehat{\boldsymbol{\mu}}_k^B = \frac{1}{\lceil n_k/2 \rceil} \sum_{j=\lfloor n_k/2 \rfloor + 1}^{n_k} \mathbf{z}_k^j \tag{4.13}$$

$$\widehat{\boldsymbol{\sigma}}_k^B = \sqrt{\frac{1}{\lceil n_k/2 \rceil - 1} \sum_{j=\lfloor n_k/2 \rfloor + 1}^{n_k} (\mathbf{z}_k^j - \widehat{\boldsymbol{\mu}}_k^B)^2} \tag{4.14}$$

comme les estimations des mêmes quantités mais calculées cette fois sur la deuxième moitié des observations. A partir de ces quatre estimateurs, il est possible de définir $\widehat{\mathbf{Pr}}\{S_k\}^A$ et $\widehat{\mathbf{Pr}}\{S_k\}^B$ en partant de la définition analytique de $\Pr\{S_k\}$ donnée dans la section 4.1. Il est à noter que bien que les quatre estimateurs $\left\{\widehat{\boldsymbol{\mu}}_k^A, \widehat{\boldsymbol{\sigma}}_k^A, \widehat{\boldsymbol{\mu}}_k^B, \widehat{\boldsymbol{\sigma}}_k^B\right\}$ aient été calculés avec seulement la moitié des observations, les valeurs $\{n_k\}$ ne sont pas divisées par deux dans le calcul de la matrice de covariance (voir équation (4.2) à la page 72). L'estimateur par *mise de côté* est alors

$$\widehat{\boldsymbol{\mu}}_g^{SPL1} = \sum_{k=1}^{K} \widehat{\mathbf{Pr}}\{S_k\}^A \cdot \widehat{\boldsymbol{\mu}}_k^B.$$

Une version plus robuste de l'estimateur $\widehat{\boldsymbol{\mu}}_g^{SPL1}$ peut être construite en tirant avantage du principe selon lequel la moyenne arithmétique de deux estimateurs non biaisés produit un nouvel estimateur avec une variance plus petite [66]. Ce nouvel estimateur peut alors être défini comme suit

$$\widehat{\boldsymbol{\mu}}_g^{SPL2} = \frac{1}{2}\left(\sum_{k=1}^{K} \widehat{\mathbf{Pr}}\{S_k\}^A \cdot \widehat{\boldsymbol{\mu}}_k^B + \sum_{k=1}^{K} \widehat{\mathbf{Pr}}\{S_k\}^B \cdot \widehat{\boldsymbol{\mu}}_k^A\right),$$

où les deux paires d'estimateurs décorrélés $(\widehat{\mathbf{Pr}}\{S_k\}^A, \widehat{\boldsymbol{\mu}}_k^B)$ et $(\widehat{\mathbf{Pr}}\{S_k\}^B, \widehat{\boldsymbol{\mu}}_k^A)$ sont utilisées dans le calcul.

4.4.4 L'estimateur par *leave-one-out*

L'approche proposée dans la section précédente est ici étendue par l'utilisation de techniques de type *leave-one-out* [17]. D est défini comme le nombre d'itérations de *leave-one-out* et l'on a $d = 1, \ldots, D$. Pour chaque alternative k, un index $\mathbf{o}_k^d$ est choisi aléatoirement dans l'ensemble $\{1, \ldots, n_k\}$ et

$$\mathbf{Z}_k(-\mathbf{o}_k^d) = \left\{\mathbf{z}_k^1, \ldots, \mathbf{z}_k^{\mathbf{o}_k^d - 1}, \mathbf{z}_k^{\mathbf{o}_k^d + 1}, \ldots, \mathbf{z}_k^{n_k}\right\}$$

est un ensemble qui contient toutes les n_k observations de l'ensemble $\mathbf{Z}_k$ à l'exception de $\mathbf{z}_k^{\mathbf{o}_k^d}$. Les K observations $\left\{\mathbf{z}_1^{\mathbf{o}_1^d}, \ldots, \mathbf{z}_K^{\mathbf{o}_K^d}\right\}$ sont maintenant

utilisées pour choisir de manière gloutonne la meilleure alternative $\mathbf{z_b}$ de la d-ième itération de *leave-one-out* où $\mathbf{b} = \arg\max_k \left\{\mathbf{z}_k^{\mathbf{o}_k^d}\right\}$. La quantité $\widehat{\boldsymbol{\mu}}_{\mathbf{b}}^{-\mathbf{o}_{\mathbf{b}}^d}$ est l'estimation de l'espérance de $\mathbf{z_b}$ calculée en prenant la moyenne arithmétique de $\mathbf{Z_b}(-\mathbf{o}_{\mathbf{b}}^d)$.

Après D itérations de *leave-one-out*, nous obtenons un ensemble

$$\left\{\widehat{\boldsymbol{\mu}}_{\mathbf{b}}^{-\mathbf{o}_{\mathbf{b}}^1}, \ldots, \widehat{\boldsymbol{\mu}}_{\mathbf{b}}^{-\mathbf{o}_{\mathbf{b}}^d}, \ldots, \widehat{\boldsymbol{\mu}}_{\mathbf{b}}^{-\mathbf{o}_{\mathbf{b}}^D}\right\}$$

de D estimations de la quantité μ_g ; chacune obtenue sur un ensemble d'observations différentes. L'estimation de μ_g par *leave-one-out* est alors

$$\widehat{\boldsymbol{\mu}}_g^{loo} = \frac{1}{D}\sum_{d=1}^{D} \widehat{\boldsymbol{\mu}}_{\mathbf{b}}^{-\mathbf{o}_{\mathbf{b}}^d}.$$

4.4.5 Expériences d'estimation de μ_g

Cette section compare expérimentalement les cinq estimateurs de μ_g sur base de sept tâches d'estimation synthétiques (voir tableau 4.4). Les problèmes d'estimation sont triés dans un ordre croissant de la valeur μ_g de manière à ce que les premières tâches soient « *faciles* » et que la dernière tâche soit « *très difficile* » . Il est intéressant de relever que la seule différence entre la tâche six et la tâche sept est que cette dernière présente un nombre de tests cent fois plus élevé sur la meilleure alternative. Le fait de tester la meilleure alternative peut donc, dans certaines circonstances, diminuer le gain espéré d'une action gloutonne. Ce phénomène – contre-intuitif – a déjà été étudié plus en détail dans la section 4.2.

Pour évaluer les performances des estimateurs, chacun des sept problèmes d'estimation a été aléatoirement échantillonné $M = 10000$ fois. Pour $m = 1, \ldots, M$, un ensemble de données $\left(\{\mathbf{Z}_k\}_{k\in\{1,\ldots,K\}}\right)_m$ est créé et utilisé pour calculer les cinq estimateurs $(\widehat{\boldsymbol{\mu}}_g)_m$. Trois mesures d'erreur sont considérées :

1. le biais

$$\textbf{biais} = MA\left(\widehat{\boldsymbol{\mu}}_g\right) - \mu_g,$$

où

$$MA\left(\widehat{\boldsymbol{\mu}}_g\right) = \frac{1}{M}\sum_{m=1}^{M}(\widehat{\boldsymbol{\mu}}_g)_m$$

2. la variance

$$\textbf{var} = \frac{1}{M-1}\sum_{m=1}^{M}\left(MA\left(\widehat{\boldsymbol{\mu}}_g\right) - (\widehat{\boldsymbol{\mu}}_g)_m\right)^2$$

	tâche 1			*tâche 2*			*tâche 3*			*tâche 4*		
	μ	σ	n	μ	σ	n	μ	σ	n	μ	σ	n
$\mathbf{z}_1$	0	0.5	5	0	1	5	0	2	5	0	4	5
$\mathbf{z}_2$	0	0.5	10	0	1	10	0	2	5	0	4	5
$\mathbf{z}_3$	0	0.5	5	0	1	5	0.5	1	10	0.5	4	5
$\mathbf{z}_4$	0.5	0.5	15	0.5	1	15	0.5	1	4	0.5	2	4
$\mathbf{z}_5$	1	0.5	10	1	1	10	0.5	1	4	0.5	3	4
$\mathbf{z}_6$							1	1	5	1	2	5
$\mathbf{z}_7$												
μ_g	0.996			0.893			0.639			0.4603		

	tâche 5			*tâche 6*			*tâche 7*		
	μ	σ	n	μ	σ	n	μ	σ	n
$\mathbf{z}_1$	0	6	5	0	8	4	0	8	4
$\mathbf{z}_2$	0	6	5	0	8	4	0	8	4
$\mathbf{z}_3$	0	6	5	0	8	4	0	8	4
$\mathbf{z}_4$	0.5	5	5	0.5	1	4	0.5	1	4
$\mathbf{z}_5$	0.5	5	5	0.5	1	4	0.5	1	4
$\mathbf{z}_6$	0.5	5	5	0.5	1	4	0.5	1	4
$\mathbf{z}_7$	1	2	5	1	1	4	1	1	400
μ_g	0.327			0.189			0.176		

TABLE 4.4 – Sept tâches d'estimation synthétiques pour évaluer les performances des cinq estimateurs de μ_g. Pour chaque tâche, trois colonnes présentent respectivement l'espérance, l'écart-type et le nombre de tests réalisés. La dernière ligne contient les valeurs de μ_g et les tâches sont triées de manière décroissante en fonction de cette valeur.

3. l'erreur quadratique moyenne (*mean square error* ou MSE)

$$\mathbf{mse} = \frac{1}{M} \sum_{m=1}^{M} \left(\mu_g - \left(\widehat{\boldsymbol{\mu}}_g \right)_m \right)^2 .$$

Tous les résultats sont reportés dans le tableau 4.5. Nous allons analyser les résultats en termes de compromis biais/variance. Pour les tâches faciles (c'est-à-dire les tâches 1 et 2), $\widehat{\boldsymbol{\mu}}_g^{max}$ et $\widehat{\boldsymbol{\mu}}_g^{PL}$ sont tous deux de bons estimateurs ayant un petit **mse**. Ceci est dû à une combinaison d'une variance et d'un biais peu élevés. Par contre, grâce à un petit biais, les estimateurs $\widehat{\boldsymbol{\mu}}_g^{SPL1}$, $\widehat{\boldsymbol{\mu}}_g^{SPL2}$ et $\widehat{\boldsymbol{\mu}}_g^{loo}$ sont efficaces sur des problèmes plus complexes et ceci malgré leur grande variance. Il est à noter que grâce au principe de combinaison d'estimateurs, la variance de $\widehat{\boldsymbol{\mu}}_g^{SPL2}$ est toujours plus faible que la variance de $\widehat{\boldsymbol{\mu}}_g^{SPL1}$. De plus, à l'exception des tâches une et deux, l'estimateur $\widehat{\boldsymbol{\mu}}_g^{loo}$ apparaît comme le plus performant.

Il ressort que, sur certains problèmes « faciles », le biais de l'estimateur naïf $\widehat{\boldsymbol{\mu}}_g^{max}$ reste modéré ce qui permet d'obtenir des estimations avec une erreur quadratique moyenne (MSE) relativement petite. Par contre, sur

	tâche 1			*tâche 2*			*tâche 3*		
	bias	**var**	**mse**	**bias**	**var**	**mse**	**bias**	**var**	**mse**
$\widehat{\boldsymbol{\mu}}_g^{max}$	0.004	0.0244	0.0244	0.138	0.081	0.100	0.6253	0.145	0.536
$\widehat{\boldsymbol{\mu}}_g^{PL}$	−0.009	0.0277	0.0278	0.0452	0.096	0.098	0.4055	0.156	0.321
$\widehat{\boldsymbol{\mu}}_g^{SPL1}$	−0.073	0.0615	0.0669	−0.223	0.223	0.273	−0.165	0.497	0.524
$\widehat{\boldsymbol{\mu}}_g^{SPL2}$	−0.065	0.0432	0.0476	−0.204	0.167	0.208	−0.147	0.328	0.350
$\widehat{\boldsymbol{\mu}}_g^{loo}$	−0.216	0.0341	0.0811	−0.335	0.070	0.182	−0.1997	0.163	0.203

	tâche 4			*tâche 5*			*tâche 6*		
	bias	**var**	**mse**	**bias**	**var**	**mse**	**bias**	**var**	**mse**
$\widehat{\boldsymbol{\mu}}_g^{max}$	1.858	1.011	4.466	3.094	2.182	11.754	3.615	6.252	19.321
$\widehat{\boldsymbol{\mu}}_g^{PL}$	1.314	1.044	2.773	2.177	2.206	6.946	2.636	6.425	13.377
$\widehat{\boldsymbol{\mu}}_g^{SPL1}$	−0.060	2.607	2.610	−0.054	5.457	5.460	−0.0004	16.711	16.71
$\widehat{\boldsymbol{\mu}}_g^{SPL2}$	−0.061	1.942	1.945	−0.063	4.212	4.215	−0.017	11.412	11.411
$\widehat{\boldsymbol{\mu}}_g^{loo}$	−0.073	1.009	1.015	−0.068	2.0240	2.028	−0.015	7.057	7.057

	tâche 7		
	bias	**var**	**mse**
$\widehat{\boldsymbol{\mu}}_g^{max}$	3.542	6.154	18.706
$\widehat{\boldsymbol{\mu}}_g^{PL}$	2.563	6.338	12.908
$\widehat{\boldsymbol{\mu}}_g^{SPL1}$	−0.073	16.740	16.744
$\widehat{\boldsymbol{\mu}}_g^{SPL2}$	−0.068	11.106	11.110
$\widehat{\boldsymbol{\mu}}_g^{loo}$	−0.061	6.908	6.911

TABLE 4.5 – Résultats des sept problèmes d'estimation synthétiques. Les quantités **bias**, **var** et **mse** sont données pour chaque tâche et pour chaque méthode d'estimation.

d'autres problèmes, c'est l'estimateur $\widehat{\boldsymbol{\mu}}_g^{loo}$ qui donne les meilleurs résultats. A partir de ces considérations, nous proposons l'estimateur suivant ($0 \leq \lambda \leq 1$)

$$\widehat{\boldsymbol{\mu}}_g^{\lambda} = \lambda \cdot \widehat{\boldsymbol{\mu}}_g^{max} + (1 - \lambda) \cdot \widehat{\boldsymbol{\mu}}_g^{loo}$$

qui combine les estimateurs $\widehat{\boldsymbol{\mu}}_g^{max}$ et $\widehat{\boldsymbol{\mu}}_g^{loo}$.

La quantité λ est donc un paramètre qui définit l'importance relative accordée à $\widehat{\boldsymbol{\mu}}_g^{max}$ et à $\widehat{\boldsymbol{\mu}}_g^{loo}$. Nous avons repris les sept tâches définies dans le tableau 4.4. Pour chaque tâche, l'erreur quadratique moyenne a été calculée pour différentes valeurs de λ dans l'intervalle $[0, 1]$ (voir figure 4.9). En fonction de la tâche, la valeur optimale du paramètre λ^* n'est pas identique. Nous n'avons pas encore défini une méthode permettant de fixer la valeur de λ en fonction du problème. Une piste pourrait être d'étudier les techniques de *shrinkage estimator* [49] qui consistent à combiner des estimateurs fortement variants mais peu biaisés avec d'autres estimateurs peu variants mais fortement biaisés. De manière générale, il semble que la valeur de λ^* varie

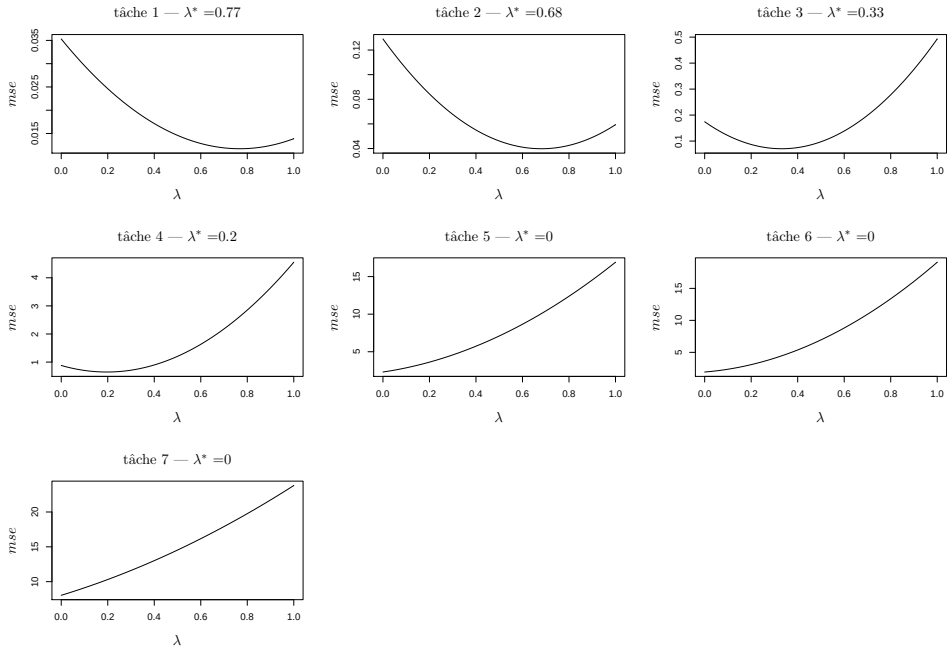


FIGURE 4.9 – Les tâches définies dans le tableau 4.4 sont réutilisées pour comparer les résultats de l'estimateur $\widehat{\boldsymbol{\mu}}_g^{\lambda}$ où le paramètre λ prend ses valeurs dans l'intervalle $[0, 1]$.

selon la difficulté du problème. Dans les premiers cas, la valeur de λ^* est élevée lorsque μ_g est élevé et λ^* ne fait ensuite que décroître lorsque la difficulté du problème augmente. Néanmoins, il est à noter que comme $\widehat{\boldsymbol{\mu}}_g^{max}$ est supérieur à $\widehat{\boldsymbol{\mu}}_g^{loo}$ dans les cas « faciles », l'erreur commise par $\widehat{\boldsymbol{\mu}}_g^{loo}$ reste très faible dans ces cas. Le gain en exactitude que l'on obtiendrait ici en utilisant des techniques de combinaison serait donc limité.

Ces estimateurs permettent maintenant d'utiliser la quantité μ_g sur des problèmes où seules les observations sont disponibles. Les chapitres 5 et 6 utiliseront ces estimateurs pour résoudre respectivement le problème de sélection de l'optimum en un minimum de tests et le *multi-armed bandit problem* (MABP).

L'estimateur par mise de côté a l'avantage de permettre de facilement estimer μ_g lorsque les valeurs $\{n_k\}$ sont différentes. Pour cela, il suffit d'utiliser d'autres valeurs de $\{n_k\}$ lors du calcul de $\widehat{\mathbf{Pr}}\{S_k\}^A$ et $\widehat{\mathbf{Pr}}\{S_k\}^B$. C'est la raison pour laquelle cet estimateur sera souvent utilisé dans la suite de cette thèse. Il permet – par exemple – de calculer une estimation de $\mu_g^{l+1}(k)$ qui est le gain espéré d'une action gloutonne à l'étape suivante si l'alternative k est testée à cette étape.

4.5 Conclusions

Dans ce chapitre, nous avons d'abord établi une définition analytique de la quantité μ_g, qui est le gain espéré d'une action d'exploitation gloutonne. Nous avons ensuite étudié l'évolution de cette quantité. Dans le cas $K = 2$, une stratégie oracle d'exploration optimale a été proposée qui garantit, à chaque étape, de tester l'alternative maximisant la quantité future de μ_g. Nous avons également montré que la quantité μ_g augmente toujours lorsque l'une des deux alternatives est testée. Enfin, nous avons expliqué que dans le cas d'une stratégie testant toujours la même alternative, l'évolution de la quantité μ_g rencontre une borne supérieure qui est inférieure à $\mu_{[K]}$. Ceci justifie formellement la technique heuristique selon laquelle l'exploitation nécessite au préalable une phase d'exploration.

Si plus de deux alternatives ($K > 2$) sont disponibles, le type de stratégie utilisé a une forte influence sur l'évolution de la quantité μ_g. Au début du problème de sélection, c'est-à-dire lorsque les écarts-types des estimateurs des moyennes ($\frac{\sigma}{\sqrt{n}}$) sont grands, tester l'alternative optimale peut diminuer la valeur de μ_g. Il est alors préférable de débuter par des techniques d'exploration et de ne passer à un mode d'exploitation gloutonne que lorsque suffisamment d'informations ont été collectées sur les différentes alternatives.

Ces observations quant à la quantité μ_g sont utilisées dans les chapitres 5 et 6 afin de proposer de nouveaux algorithmes résolvant respectivement le problème de sélection de l'optimum en un minimum de tests (section 3.2 à la page 43) et du *multi-armed bandit problem* (section 3.3 à la page 58).

Ce chapitre se termine en proposant une série d'estimateurs de la quantité μ_g. Il est à noter que contrairement aux méthodes présentées dans la section 3.2 nous n'allons pas proposer d'algorithmes qui garantissent une borne sur $\Pr\{S_{[K]}\}$ mais qui utiliseront plutôt une estimation de μ_g.

Chapitre 5

Les stratégies de sélection de l'optimum – contributions

Dans ce chapitre, nous considérons des problèmes de sélection multivariée de type *selecting the best*. Il s'agit d'élaborer des stratégies qui échantillonnent les différentes alternatives dans le but de repérer, avec un budget de tests limité et après la phase d'échantillonnage, la variable aléatoire ayant la plus grande espérance.

Dans la section 3.2, nous avons vu que les différentes stratégies d'échantillonnage proposées dans la littérature demandent toutes, d'une manière ou d'une autre, à l'utilisateur de fixer une borne à la quantité $\Pr\{S_{[K]}\}$. Cette quantité correspond à la probabilité que l'alternative ayant la plus grande moyenne arithmétique soit également celle présentant la plus grande espérance.

Nous allons proposer trois nouvelles stratégies : optiK2Estim, *PCSsearch* et exploreMaxμ_g. Plutôt que de définir des bornes sur $\Pr\{S_{[K]}\}$, ces nouvelles stratégies vont chacune – à leur manière – chercher à maximiser la quantité $\Pr\{S_{[K]}\}$ en utilisant des méthodes d'estimation. Il est à noter que ces trois stratégies font toutes l'hypothèse que les variables associées aux alternatives suivent des distributions normales :

- La stratégie optiK2Estim est une stratégie spécifique dans le cas où seules deux alternatives sont en compétition. Elle utilise les résultats de la section 4.2.1 où nous avons vu que pour maximiser l'augmentation de $\Pr\{S_{[K]}\}$ si $K = 2$, il faut sélectionner l'alternative qui maximise la diminution de la quantité $\frac{\sigma^2_{[K]}}{n_{[K]}} + \frac{\sigma^2_{[K-1]}}{n_{[K-1]}}$. Vu la non-disponibilité de $\sigma^2_{[K]}$ et de $\sigma^2_{[K-1]}$, ce sont les estimations des variances des deux alternatives qui sont utilisées.
- A chaque étape de la stratégie *PCSsearch* et pour toutes les alternatives k, la valeur de la quantité $\Pr\left\{S^{l+1}_{[K]}(k)\right\}$ est estimée où $S^{l+1}_{[K]}(k)$ correspond à l'événement « *faire une sélection correcte à l'étape* $l+1$ *si*

c'est l'alternative k qui est testée l'étape l ≫ . La stratégie *PCSsearch* consiste ensuite à sélectionner l'alternative qui maximise l'estimation de la quantité $\Pr\left\{S_{[K]}^{l+1}(k)\right\}$.

– La stratégie exploreMaxμ_g utilise les conclusions tirées de l'étude (voir section 4.2.2) de l'évolution de μ_g lorsque le nombre de tests sur les alternatives augmente. Sur base d'*heuristiques*, cette stratégie va tester les alternatives en cherchant à éviter les situations qui font diminuer la valeur de μ_g.

Les sélections réalisées dans ce chapitre doivent se faire dans un environnement incertain. C'est un exemple de problème où l'information sur l'état des alternatives est imparfaite (en anglais *Imperfect State Information* (ISI) [12]). L'approche adoptée pour présenter les trois algorithmes consiste à résoudre d'abord les problèmes lorsque l'information sur les alternatives est parfaite (en anglais *Perfect State Information* (PSI) [12]) et ensuite d'utiliser des méthodes d'estimation sur les variables inconnues. C'est une approche identique à celle utilisée pour résoudre certains problèmes de programmation dynamique linéaire (*certainty equivalence*) [12].

Ce chapitre contient les contributions de cette thèse en rapport avec le problème de la sélection de l'optimum avec un budget de tests limité. Il est structuré comme suit : les trois prochaines sections présentent respectivement les algorithmes optiK2Estim, *PCSsearch* et exploreMaxμ_g. Dans la section 5.4, des études expérimentales montrent que nos approches sont compétitives avec d'autres stratégies existant dans la littérature. Les expériences sont réalisées à la fois sur des données créées de manière synthétique et sur des données réelles. Ces données réelles portent sur les problèmes de sélection de modèles locaux (voir section 2.6.2.2 à la page 37). La dernière section contient les conclusions de ce chapitre.

5.1 La stratégie optiK2Estim ($K = 2$)

Le théorème 4.4 a défini les conditions d'exploration optimale quand seules deux alternatives sont en compétition. Ceci nous a permis de définir la stratégie optiK2. Le principe d'optiK2 est de toujours faire augmenter la quantité $\Pr\{S_{[K]}\}$ en diminuant

$$\mathrm{Var}\left[\widehat{\boldsymbol{\mu}}_{k_1} - \widehat{\boldsymbol{\mu}}_{k_2}\right] = \frac{\sigma_{k_1}^2}{n_{k_1}} + \frac{\sigma_{k_2}^2}{n_{k_2}}$$

où k_1 et k_2 sont les indices des deux alternatives. A chaque étape, cette stratégie calcule la valeur de N_Δ (voir équation (4.7) à la page 80) à partir des variances des deux alternatives et du nombre de tests réalisés sur chacune d'elles. Le signe de N_Δ est ensuite utilisé pour choisir l'alternative à tester.

La stratégie optiK2 fait l'hypothèse que les variances sont connues (PSI). Or elles ne le sont pas ; nous proposons dès lors de remplacer ces variances

Algorithme 13 optiK2Estim

1: Tester chaque alternative I fois où $I \geq 2$.
2: **boucler**
3: $\quad \mathbf{N}_\Delta = n_{k_1} \cdot (n_{k_1} + 1) \cdot \left(\widehat{\boldsymbol{\sigma}}^2_{k_2} - \widehat{\boldsymbol{\sigma}}^2_{k_1} \right) + \widehat{\boldsymbol{\sigma}}^2_{k_1} \cdot (n_{k_2} + n_{k_1} + 1) \cdot (n_{k_1} - n_{k_2})$.
4: $\quad$ **si** $\mathbf{N}_\Delta < 0$ **alors**
5: $\quad\quad \widehat{\mathbf{k}} \leftarrow k_1$.
6: $\quad$ **sinon**
7: $\quad\quad \widehat{\mathbf{k}} \leftarrow k_2$.
8: $\quad$ **fin si**
9: $\quad$ Tester $\mathbf{z}_{\widehat{\mathbf{k}}}$ et mettre à jour les variables utilisées en 3.
10: **fin boucle**

par leurs estimations (ISI).

La nouvelle stratégie est dénotée optiK2Estim et est présentée dans l'algorithme 13. Chaque alternative est d'abord testée I fois (ligne 1) ce qui permet d'obtenir les observations nécessaires au calcul des premières estimations des variables des deux alternatives. Ensuite, à chaque étape, la quantité $\mathbf{N}_\Delta$ est calculée (ligne 3) et en fonction de son signe, la stratégie choisit l'alternative à tester où $\widehat{\mathbf{k}}$ désigne son indice.

D'un point de vue computationnel, cet algorithme est très rapide car il ne demande à chaque étape que le calcul de deux variances.

Il est intéressant de noter qu'optiK2Estim se base uniquement sur les estimations des variances ($\widehat{\boldsymbol{\sigma}}^2$) et sur le nombre de tests réalisés (n) pour choisir l'alternative à tester. Dans le cas où seules deux alternatives sont en compétition ($K = 2$), la stratégie optiK2Estim ne tient donc pas compte des moyennes arithmétiques pour prendre sa décision.

Une validation expérimentale de l'algorithme 13 est présentée en section 5.4.1.3.

5.2 La stratégie *PCSsearch* ($K \geq 2$)

Cette section présente la stratégie d'exploration *PCSsearch*. Le principe de cet algorithme consiste à faire augmenter, à chaque étape, la quantité $\Pr\{S_{[K]}\}$ de la manière la plus forte possible.

La notion de PCS (section 3.2) a été proposée dans la littérature Monte Carlo pour résoudre les problèmes de sélection de la meilleure alternative à l'aide de simulateurs. Ici, nous proposons d'utiliser cette notion comme mesure de l'efficacité d'une action d'exploration.

Pour présenter notre stratégie, nous supposons être dans un cas où les caractéristiques des K alternatives sont connues (PSI).

Soit l l'étape actuelle. La stratégie *PCSsearch* calcule pour chaque alter-

Algorithme 14 *PCSsearch*

1: Tester chaque alternative I fois où $I \geq 2$.
2: **boucler**
3: **pour** $k = 1$ à K **faire**
4: $n_{\#}^{l+1} \leftarrow \left\{n_1^l, \ldots, n_k^l + 1, \ldots, n_K^l\right\}$.
5: $\mathbf{PCS}_k^{l+1} \leftarrow calculerPCS\left(\left\{\widehat{\boldsymbol{\mu}}_k\right\}_{k\in\{1,K\}}, \left\{\widehat{\boldsymbol{\sigma}}_k\right\}_{k\in\{1,K\}}, n_{\#}^{l+1}, [\widehat{\mathbf{K}}]\right)$.
6: **fin pour**
7: $\widehat{\mathbf{k}} \leftarrow \arg\max_k \mathbf{PCS}_k^{l+1}$.
8: Tester $\mathbf{z}_{\widehat{\mathbf{k}}}$ et mettre à jour les variables utilisées en 5.
9: **fin boucle**

native k la quantité

$$\Pr\left\{S_{[K]}^{l+1}(k)\right\}$$

qui correspond à la probabilité de sélectionner l'alternative optimal $[K]$ à l'étape $l+1$ si l'alternative k est testée à l'étape l. La stratégie sélectionne ensuite l'alternative qui maximise la probabilité de faire une sélection correcte à l'étape $l+1$.

Afin de calculer la probabilité de faire une sélection correcte, la stratégie utilise la fonction $calculerPCS$ qui se base sur les résultats du théorème 4.1 à la page 71. La fonction $calculerPCS$ possède quatre paramètres. Les deux premiers sont les caractéristiques des alternatives (μ, σ) et les deux derniers paramètres sont le nombre de tests réalisés (n) et l'indice de l'alternative optimale ($[K]$).

Par définition, les caractéristiques des K alternatives sont bien entendu inconnues. Nous proposons alors d'appliquer le principe de *plug-in* en remplaçant dans le calcul de la $\Pr\{S_{[K]}\}$ les valeurs inconnues $\{\mu_k\}$ et $\{\sigma_k\}$ par leurs estimations $\{\widehat{\boldsymbol{\mu}}_k\}$ et $\{\widehat{\boldsymbol{\sigma}}_k\}$.

Cette stratégie est décrite dans l'algorithme 14. Chaque alternative est tout d'abord testée I fois (ligne 1), ce qui permet d'avoir une connaissance a priori sur l'environnement dans lequel les décisions doivent être prises. Pour chaque alternative, l'algorithme *PCSsearch* simule un test sur celle-ci en augmentant de un le nombre de tests réalisés (ligne 4). L'algorithme calcule ensuite la $\Pr\{S_{[K]}\}$ future de l'étape suivante $l+1$ si l'alternative k était testée à l'étape courante l (ligne 5) et celle qui augmente le plus cette probabilité future de faire une sélection correcte est sélectionnée pour être testée (ligne 7 et 8).

Il est à noter que sur un total de H étapes, la fonction $calculerPCS$ est utilisée $K \times H$ fois. Or, celle-ci utilise la méthode de Genz, qui a une complexité en $O(K \times R_{Genz})$ (voir section 4.3), donc la complexité de *PCSsearch* est en $O(K^2 \times H \times R_{Genz})$.

Algorithme 15 La stratégie exploreMaxμ_g – version PSI

1: **boucler**
2: **si** $\mu_g \leq \mu_g^{l+1}([K])$ **alors**
3: $\widehat{\mathbf{k}} \leftarrow [K]$ # Sélection oracle gloutonne
4: **sinon**
5: $\widehat{\mathbf{k}} \leftarrow$ sélection aléatoire où p_k est la probabilité de choisir k.
6: **fin si**
7: Tester $\mathbf{z}_{\widehat{\mathbf{k}}}$
8: **fin boucle**

5.3 La stratégie exploreMaxμ_g ($K \geq 2$)

Cette section propose une alternative à *PCSsearch*, dénotée exploreMaxμ_g, pour résoudre le problème de sélection de la meilleure alternative avec un budget de tests limité. Cet algorithme, à la différence de *PCSsearch*, cherche à toujours maximiser μ_g. Il est à noter que μ_g atteint son maximum (qui vaut $\mu_{[K]}$) uniquement lorsque $\Pr\{S_{[K]}\} = 1$ et donc que maximiser μ_g constitue une manière indirecte d'optimiser $\Pr\{S_{[K]}\}$.

La section 4.2 a montré que, dans certains cas, tester une alternative peut diminuer la valeur de μ_g lorsque $K > 2$. Notre algorithme va chercher à éviter ce genre de situation et visera une augmentation de μ_g à chaque étape.

Nous définissons la quantité $\mu_g^{l+1}(k)$ comme le gain espéré d'une action gloutonne à l'étape $l+1$ si l'alternative k est testée à l'étape l. L'algorithme se base sur la considération suivante. Lorsque l'on observe une décroissance de la quantité μ_g après une sélection gloutonne oracle $\left(\mu_g^l > \mu_g^{l+1}([K])\right)$, cela signifie que nous sommes dans des conditions défavorables. Dans la section 4.2.2, nous avons vu que de telles situations défavorables se produisent lorsque les variances des moyennes arithmétiques (σ_k^2/n_k) des mauvaises alternatives sont grandes. Avant de réaliser une sélection gloutonne, il faut dans ce cas tester les alternatives sub-optimales ayant une valeur σ_k^2/n_k élevée.

Nous commençons par décrire l'algorithme exploreMaxμ_g dans une version PSI qui suppose que les caractéristiques des distributions des variables associées aux alternatives sont connues. Pour la procédure de sélection, nous proposons d'utiliser la méthode suivante (voir algorithme 15) : si l'on observe que $\mu_g \leq \mu_g^{l+1}([K])$ alors on teste l'optimum $[K]$ (ligne 3), sinon la quantité p_k est utilisée comme la probabilité de sélectionner l'alternative k (ligne 5) où nous définissons

$$p_k = \frac{\sigma_k^2/n_k}{\sum_{\bar{k}=1}^{K} \sigma_{\bar{k}}^2/n_{\bar{k}}}$$

comme une mesure de la variance relative de $\widehat{\boldsymbol{\mu}}_k$ comparée à la variance

Algorithme 16 La stratégie exploreMaxμ_g

1: Tester chaque alternative I fois où $I \geq 2$.
2: **boucler**
3: Calculer les estimateurs $\widehat{\boldsymbol{\mu}}_g$ et $\widehat{\boldsymbol{\mu}}_g^{l+1}\Big([\widehat{\mathbf{K}}]\Big)$.
4: Pour chaque alternative k, calculer la borne supérieure de l'estimateur de la variance $UB(\widehat{\boldsymbol{\sigma}}_k^2)$ et calculer ensuite $\mathbf{p}_k$
5: **si** $\widehat{\boldsymbol{\mu}}_g \leq \widehat{\boldsymbol{\mu}}_g^{l+1}\Big([\widehat{\mathbf{K}}]\Big)$ et $\mathbf{p}_{[\widehat{\mathbf{K}}]} \geq G$ **alors**
6: $\widehat{\mathbf{k}} \leftarrow [\widehat{\mathbf{K}}]$ # Sélection gloutonne
7: **sinon**
8: $\widehat{\mathbf{k}} \leftarrow$ sélection aléatoire où $\mathbf{p}_k$ est la probabilité de choisir k.
9: **fin si**
10: Tester $\mathbf{z}_{\widehat{\mathbf{k}}}$ et mettre à jour les différentes variables utilisées.
11: **fin boucle**

totale de toutes les moyennes arithmétiques.

Le principe de cet algorithme est de commencer par vérifier que les conditions d'exécution d'une action gloutonne sont remplies. Si oui, l'action gloutonne est exécutée. Dans le cas contraire, c'est une exploration qui est exécutée où les variables ayant une valeur σ^2/n élevée ont de fortes chances d'être sélectionnées.

Les caractéristiques des alternatives sont – en réalité – inaccessibles et nous allons alors utiliser des techniques d'estimation (ISI). La stratégie qui en découle est décrite dans l'algorithme 16.

A la ligne 3, $\widehat{\boldsymbol{\mu}}_g$ et $\widehat{\boldsymbol{\mu}}_g^{l+1}\Big([\widehat{\mathbf{K}}]\Big)$ sont calculés en utilisant l'estimateur par *mise de côté* décrit dans la section 4.4.3.

L'indice $[\widehat{\mathbf{K}}]$ correspond à l'indice de l'alternative ayant la plus grande moyenne arithmétique et $\widehat{\boldsymbol{\mu}}_g^{l+1}\Big([\widehat{\mathbf{K}}]\Big)$ représente donc l'estimation de la prochaine valeur de μ_g si une action gloutonne est réalisée. Il faut noter que contrairement à la version PSI de l'algorithme 15, la sélection n'est plus une sélection *oracle* gloutonne et il se peut que $[\widehat{\mathbf{K}}]$ ne constitue pas l'optimum.

Soit

$$UB(\widehat{\boldsymbol{\sigma}}_k^2) = (n_k - 1)\,\frac{\widehat{\boldsymbol{\sigma}}_k^2}{\chi^2_{\alpha_{ub}/2}}$$

une borne optimiste de l'estimation de la variance de $\mathbf{z}_k$ où $\chi^2_{\alpha_{ub}/2}$ est le quantile d'ordre $\alpha_{ub}/2$ de la loi χ^2 avec $n_k - 1$ degré de liberté [1].

Soit

$$\mathbf{p}_k = \frac{UB(\widehat{\boldsymbol{\sigma}}_k^2)/n_k}{\sum_{\bar{k}=1}^{K} UB(\widehat{\boldsymbol{\sigma}}_{\bar{k}}^2)/n_{\bar{k}}}.$$

1. Dans les expériences, α_{ub} a été arbitrairement fixé à 0.01. Il est à noter que l'on utilise la borne supérieure $UB(\widehat{\boldsymbol{\sigma}}_k^2)$ pour forcer l'algorithme – surtout au début – à tester les alternatives peu explorées.

une quantité qui reporte l'estimation optimiste de variance de la moyenne arithmétique de l'alternative k sur l'estimation optimiste de la variance totale de toutes les alternatives. La quantité $\mathbf{p}_k$ donne la proportion relative de la variance de $\widehat{\boldsymbol{\mu}}_k$ comparée à la variance totale des K moyennes arithmétiques.

A la ligne 5 de l'algorithme 16, deux tests sont exécutés. On vérifie si $\widehat{\boldsymbol{\mu}}_g \leq \widehat{\boldsymbol{\mu}}_g^{l+1}\Big([\widehat{\mathbf{K}}]\Big)$ et on réalise un deuxième test $\Big(\mathbf{p}_{[\widehat{\mathbf{K}}]} \geq G\Big)$ qui vérifie que la variance relative de $\widehat{\boldsymbol{\mu}}_{[\widehat{\mathbf{K}}]}$ soit supérieure à une valeur G. Ces deux tests déterminent s'il faut réaliser une action d'exploration ou une action d'exploitation. Considérons les deux cas suivants :

1. la quantité $[\widehat{\mathbf{K}}]$ ne désigne pas l'alternative optimale ($[\widehat{\mathbf{K}}] \neq [K]$).
   - Si $\widehat{\boldsymbol{\mu}}_g \leq \widehat{\boldsymbol{\mu}}_g^{l+1}\Big([\widehat{\mathbf{K}}]\Big)$ est *vrai* et $\Big(\mathbf{p}_{[\widehat{\mathbf{K}}]} \geq G\Big)$ est *vrai*.
     Comme $[\widehat{\mathbf{K}}]$ n'est pas l'alternative optimale et que $\sigma^2_{[\widehat{\mathbf{K}}]}/n_{[\widehat{\mathbf{K}}]}$ est relativement grand, cela signifie qu'avec la grande variance de sa moyenne arithmétique, l'alternative $[\widehat{\mathbf{K}}]$ a eu de la chance d'être sélectionnée – par erreur – comme étant l'optimum. Il faut donc diminuer la variance de la moyenne arithmétique de l'alternative $[\widehat{\mathbf{K}}]$. Tester cette alternative permettra à la stratégie exploreMaxμ_g de réaliser – en faisant diminuer la variance de $\widehat{\boldsymbol{\mu}}_{[\widehat{\mathbf{K}}]}$ – que $[\widehat{\mathbf{K}}]$ ne constitue pas l'optimum.
   - Si $\widehat{\boldsymbol{\mu}}_g \leq \widehat{\boldsymbol{\mu}}_g^{l+1}\Big([\widehat{\mathbf{K}}]\Big)$ est *vrai* et $\Big(\mathbf{p}_{[\widehat{\mathbf{K}}]} \geq G\Big)$ est *faux* [2].
     Si – en comparaison des autres alternatives – la quantité σ^2/n de la variable $[\widehat{\mathbf{K}}]$ est très petite, cela signifie qu'elle a été sélectionnée car les autres alternatives (qui ont une variance plus grande) n'ont pas eu de chance et ont généré de mauvais résultats lorsqu'elles ont été testées. Dans ce cas, il ne faut pas tester l'alternative $[\widehat{\mathbf{K}}]$ – car $\sigma^2_{[\widehat{\mathbf{K}}]}/n_{[\widehat{\mathbf{K}}]}$ est déjà peu élevé – mais il faut forcer exploreMaxμ_g à effectuer de l'exploration. C'est le rôle du deuxième test de la ligne 5 où $\mathbf{p}_{[\widehat{\mathbf{K}}]}$ mesure la proportion relative de la variance de $\widehat{\boldsymbol{\mu}}_{[\widehat{\mathbf{K}}]}$ comparée à la variance totale. Si $\mathbf{p}_{[\widehat{\mathbf{K}}]}$ est plus petit que G alors exploreMaxμ_g est forcé d'explorer les alternatives [3].
   - Si $\widehat{\boldsymbol{\mu}}_g \leq \widehat{\boldsymbol{\mu}}_g^{l+1}\Big([\widehat{\mathbf{K}}]\Big)$ est *faux* et $\Big(\mathbf{p}_{[\widehat{\mathbf{K}}]} \geq G\Big)$ est *vrai*.
     Tester $[\widehat{\mathbf{K}}]$ est un mauvais choix car cela fera diminuer μ_g et c'est ce que exploreMaxμ_g cherche à éviter. L'algorithme va alors explorer les alternatives.
   - Si $\widehat{\boldsymbol{\mu}}_g \leq \widehat{\boldsymbol{\mu}}_g^{l+1}\Big([\widehat{\mathbf{K}}]\Big)$ est *faux* et $\Big(\mathbf{p}_{[\widehat{\mathbf{K}}]} \geq G\Big)$ est *faux*.
     Même situation que précédemment où – pour éviter de faire dimi-

2. C'est pour ce cas que le deuxième test de la ligne 5 a été introduit.
3. Dans les expériences, G a été arbitrairement fixé à $1/(10 \cdot K)$. Cette valeur permet à G de rester faible par rapport aux valeurs prises par $\mathbf{p}_{[\widehat{\mathbf{K}}]}$ lorsque K augmente.

nuer μ_g – il est préférable d'explorer les alternatives.

2. la quantité $[\widehat{\mathbf{K}}]$ désigne l'alternative optimale ($[\widehat{\mathbf{K}}] = [K]$).
   - Si $\widehat{\boldsymbol{\mu}}_g \leq \widehat{\boldsymbol{\mu}}_g^{l+1}\Big([\widehat{\mathbf{K}}]\Big)$ est *vrai* et $\Big(\mathbf{p}_{[\widehat{\mathbf{K}}]} \geq G\Big)$ est *vrai*.
     L'algorithme est dans une situation favorable où tester l'optimal $[K]$ fera augmenter μ_g et $[K]$ sera donc sélectionné à la ligne 6.
   - Si $\widehat{\boldsymbol{\mu}}_g \leq \widehat{\boldsymbol{\mu}}_g^{l+1}\Big([\widehat{\mathbf{K}}]\Big)$ est *vrai* et $\Big(\mathbf{p}_{[\widehat{\mathbf{K}}]} \geq G\Big)$ est *faux*.
     L'algorithme est encore dans une situation où tester l'optimal $[K]$ fera augmenter μ_g mais comme la variance de $\mu_{[\widehat{\mathbf{K}}]}$ est relativement petite, l'algorithme va plutôt explorer les alternatives.
   - Si $\widehat{\boldsymbol{\mu}}_g \leq \widehat{\boldsymbol{\mu}}_g^{l+1}\Big([\widehat{\mathbf{K}}]\Big)$ est *faux* et $\Big(\mathbf{p}_{[\widehat{\mathbf{K}}]} \geq G\Big)$ est *vrai*.
     Tester l'alternative optimale $[K]$ fera diminuer μ_g. Comme c'est le type de situation que cherche a éviter la stratégie exploreMaxμ_g, l'algorithme va explorer les alternatives.
   - Si $\widehat{\boldsymbol{\mu}}_g \leq \widehat{\boldsymbol{\mu}}_g^{l+1}\Big([\widehat{\mathbf{K}}]\Big)$ est *faux* et $\Big(\mathbf{p}_{[\widehat{\mathbf{K}}]} \geq G\Big)$ est *faux*.
     Encore une fois, tester l'alternative optimale $[K]$ fera diminuer μ_g et l'algorithme va donc plutôt explorer les alternatives.

Dans les différents cas, l'algorithme va donc toujours chercher à réaliser l'action (exploration/exploitation) adéquate.

A la ligne 3 de l'algorithme 16, les valeurs $\widehat{\boldsymbol{\mu}}_g$ et $\widehat{\boldsymbol{\mu}}_g^{l+1}\Big([\widehat{\mathbf{K}}]\Big)$ sont calculées via la méthode de *mise de côté* (voir section 4.4.3). Chaque utilisation de cette méthode utilise $2 \times K$ fois l'approche quasi-Monte Carlo de Genz (voir section 4.3). Sur un total de H étapes, la méthode de Genz – de complexité $O(K \times R_{Genz})$ – est donc utilisée $4 \times K \times H$ fois.

5.4 Expérimentations et discussion

Cette section évalue les performances des stratégies optiK2Estim, *PCSsearch* et exploreMaxμ_g en les comparant avec d'autres stratégies d'exploration présentées dans la section 3.2. Les tests sont réalisés en deux parties. Dans un premier temps, les stratégies sont comparées sur des données créées de manière synthétique à partir de distributions gaussiennes et, dans un second temps, les stratégies sont comparées à des problèmes de sélection de modèles locaux construits à partir de données réelles.

Dans toutes les expériences, les stratégies commencent toujours par réaliser $I = 5$ tests sur chaque alternative.

5.4.1 Problèmes créés de manière synthétique

Cette section compare l'efficacité des diverses stratégies sur des données synthétiques et se divise en trois parties. Dans un premier temps, les stratégies *PCSsearch* et exploreMaxμ_g sont comparées à une version de la stratégie

	E1	E2	E3	E4	E5	E6
K	3	3	5	5	10	10
σ^L	2	0	2	0	2	0
σ^U	3	5	3	5	3	5

TABLE 5.1 – Les six problèmes synthétiques sont numérotés de E1 à E6 et se distinguent par le nombre d'alternatives K et les écarts-types minimum σ^L et maximum σ^U.

d'exploration $\mathcal{S}+\mathcal{R}$ où $\alpha_g = 0.2$ (voir section 3.2.3). Ensuite, nos deux stratégies sont comparées avec quatre versions de l'algorithme F-race où $\alpha_f = \{0.4, 0.2, 0.1, 0.05\}$ (voir section 3.2.4.3). Enfin, la troisième partie de cette section présente des tests réalisés au moyen de la stratégie optiK2Estim dans le cas où seules deux alternatives sont en compétition.

Pour comparer les performances des nouvelles stratégies aux stratégies $\mathcal{S}+\mathcal{R}$ et F-race , six problèmes vont être créés de manière synthétique. Chaque problème est composé de cinq cents tâches générées de manière aléatoire. Ces tâches sont obtenues en échantillonnant de manière uniforme les moyennes ($\widehat{\boldsymbol{\mu}}_k \sim U[0,1]$) et les écarts-types ($\widehat{\boldsymbol{\sigma}}_k \sim U[\sigma^L, \sigma^U]$) où σ^L et σ^U sont repris dans le tableau 5.1. Pour chaque tâche, le gain obtenu après avoir testé une alternative k suit une distribution normale ($\mathbf{z}_k \sim N[\mu_k, \sigma_k]$). Les problèmes se distinguent par le nombre d'alternatives et par les valeurs de σ^L et de σ^U.

A la fin de chacune des cinq cents tâches, chaque stratégie renvoie l'indice de l'alternative sélectionnée comme l'optimum et le score d'une stratégie pour cette tâche est l'espérance de l'alternative en question. A la fin des cinq cents tâches, et pour chacune des stratégies, on a donc un vecteur contenant les cinq cents espérances des alternatives sélectionnées. Ce sont ces vecteurs qui seront ensuite utilisés pour comparer (via des t-tests jumelés) les performances des différentes stratégies.

5.4.1.1 *PCSsearch* et exploreMaxμ_g contre $\mathcal{S}+\mathcal{R}$

Dans cette section, les stratégies *PCSsearch* et exploreMaxμ_g sont comparées à une version de l'algorithme $\mathcal{S}+\mathcal{R}$ où $\alpha_g = 0.2$ (voir section 3.2.3). Le nombre d'étapes (H) est fixé par $\mathcal{S}+\mathcal{R}$ et nos deux stratégies devront réaliser l'optimisation en un nombre strictement identique d'étapes.

Les résultats sont repris dans le tableau 5.2. Le chiffre dans la ligne L et la colonne C correspond au nombre de fois que la stratégie de la ligne L a significativement battu ($p < 0.05$) la stratégie de la colonne C. Pour rappel, il y a au total six problèmes (voir tableau 5.1)

Nous observons que sur les six problèmes, $\mathcal{S}+\mathcal{R}$ ne s'est jamais montré significativement supérieur à *PCSsearch* et exploreMaxμ_g. Par contre $\mathcal{S}+\mathcal{R}$ se fait battre significativement trois fois par les deux autres stratégies. Nous

	PCSsearch	exploreMaxμ_g	$\mathcal{S}+\mathcal{R}(\alpha_g = 0.2)$
PCSsearch		0	3
exploreMaxμ_g	3		3
$\mathcal{S}+\mathcal{R}(\alpha_g = 0.2)$	0	0	

TABLE 5.2 – Résultats expérimentaux lorsque la stratégie $\mathcal{S}+\mathcal{R}$ est comparée avec les stratégies *PCSsearch* et exploreMaxμ_g sur *six problèmes* générés de manière artificielle.

observons également que *PCSsearch* est trois fois significativement inférieur à exploreMaxμ_g et que exploreMaxμ_g ne se fait jamais battre significativement ni par $\mathcal{S}+\mathcal{R}$ ni par *PCSsearch*.

5.4.1.2 *PCSsearch* et exploreMaxμ_g contre F-race

Dans le cas des stratégies F-race (voir section 3.2.4.3), il est possible de fixer un nombre d'étapes maximum (H_{max}). Dans cette section, H_{max} va prendre les valeurs 2000, 1000, 500 et 200. Le nombre de problèmes (voir tableau 5.1) est donc multiplié par quatre, soit un total de *vingt-quatre problèmes*.

Durant les expériences, nous allons utiliser la version de *F-race* implémentée dans la fonction *race* de la librairie du même nom [4] du langage de programmation R [5].

Les tests sont réalisés en quatre groupes de trois stratégies. *PCSsearch* et exploreMaxμ_g sont, chaque fois, comparés avec une des quatre versions de F-race ($\alpha_f = \{0.4, 0.2, 0.1, 0.05\}$). La stratégie F-race ne nécessite pas toujours l'utilisation des H_{max} étapes pour déterminer l'alternative optimale. Le nombre d'étapes (H) est fixé par F-race et les stratégies *PCSsearch* et exploreMaxμ_g devront réaliser l'optimisation en ce même nombre d'étapes.

Les résultats pour chacune des quatre versions de F-race sont repris dans le tableau 5.3. L'annexe A contient les mêmes résultats mais présentés de manière détaillée. Dans le tableau 5.3, nous observons que lorsque $\alpha_f = 0.05$, la stratégie F-race est treize fois significativement inférieure à exploreMaxμ_g et trois fois significativement inférieure à *PCSsearch*. Comme la stratégie ne dispose que d'un budget de H_{max} rounds pour procéder à la sélection, la faible performance de la stratégie F-race lorsque $\alpha_f = 0.05$ s'explique en partie par le fait qu'elle n'aura pas toujours eu la possibilité (surtout lorsque $H_{max} = 200$) d'éliminer les alternatives sous-optimales et d'ainsi concentrer sa puissance de calcul sur les tests des meilleures alternatives. Les résultats détaillés de l'annexe A confirment cette remarque. Lorsque $H_{max} = 200$, F-race ($\alpha_f = 0.05$) est cinq fois sur six inférieur à

4. `http://cran.r-project.org/web/packages/race/index.html`
5. `http://www.r-project.org/`

	PCSsearch	exploreMaxμ_g	F-race ($\alpha_f = 0.05$)
PCSsearch		0	3
exploreMaxμ_g	10		13
F-race ($\alpha_f = 0.05$)	1	0	

	PCSsearch	exploreMaxμ_g	F-race ($\alpha_f = 0.1$)
PCSsearch		0	2
exploreMaxμ_g	9		8
F-race ($\alpha_f = 0.1$)	5	0	

	PCSsearch	exploreMaxμ_g	F-race ($\alpha_f = 0.2$)
PCSsearch		0	2
exploreMaxμ_g	10		3
F-race ($\alpha_f = 0.2$)	10	0	

	PCSsearch	exploreMaxμ_g	F-race ($\alpha_f = 0.4$)
PCSsearch		0	0
exploreMaxμ_g	8		0
F-race ($\alpha_f = 0.2$)	9	2	

TABLE 5.3 – Résultats expérimentaux lorsque trois versions de la stratégie F-race sont comparées avec les stratégies *PCSsearch* et exploreMaxμ_g sur *vingt-quatre problèmes* générés de manière artificielle.

exploreMaxμ_g par contre lorsque $H_{max} = 2000$, F-race ($\alpha_f = 0.05$) devient seulement deux fois sur six inférieur à exploreMaxμ_g (voir annexe A).

Nous observons d'ailleurs que lorsque la valeur de α_f augmente, F-race se fait moins souvent surpasser par les deux autres stratégies. Avec une valeur de $\alpha_f = 0.2$, les performances de F-race deviennent très proches de celles de exploreMaxμ_g et lorsque $\alpha_f = 0.4$, F-race surpasse exploreMaxμ_g deux fois sur vingt-quatre.

Nous observons les bonnes performances générales d'exploreMaxμ_g qui a été significativement inférieur aux deux autres stratégies seulement deux fois sur un total de 192 problèmes [6] et ceci sans ajuster de paramètres (contrairement – par exemple – à F-race où le paramètre α_f doit être ajusté).

Pour chaque valeur de α_f dans le tableau 5.3 les stratégies ont été comparées sur vingt-quatre problèmes où chaque problème est composé de cinq cents tâches. Ceci signifie que pour chaque valeur de α_f, les stratégies ont été testées sur 12000 tâches (24×500). Pour rappel, à la fin de chaque tâche, un

6. Il y a six problèmes différents (E1,...,E6), quatre valeurs de H_{max} (200, 500, 1000, 2000), quatre valeurs de α_f (0.05, 0.1, 0.2, 0.4) et deux algorithmes en compétition (*PCSsearch* et F-race) donc $6 \times 4 \times 4 \times 2 = 192$.

	T1	T2	T3
σ^L	2	0	0
σ^U	3	5	7

TABLE 5.4 – Les trois problèmes synthétiques se distinguent par les écarts-types minimum σ^L et maximum σ^U. Le nombre d'alternatives K est toujours fixé à deux.

score a été obtenu qui correspond à l'espérance de l'alternative sélectionnée. Différents vecteurs contenant les scores des stratégies sur les 12000 tâches ont été construits. Nous avons ensuite comparé via des t-tests jumelés les différents vecteurs et avons observé que :

- *lorsque* $\alpha_f = 0.05$, exploreMaxμ_g est significativement supérieur aux deux autres stratégies ($p << 0.05$) alors qu'il n'y a pas de différence statistique suffisante entre les performances de *PCSsearch* et F-race ($p = 0.0617$),
- *lorsque* $\alpha_f = 0.1$, exploreMaxμ_g est encore toujours significativement supérieur aux deux autres stratégies ($p << 0.05$) et il n'y a toujours pas de différence statistiquement significative entre *PCSsearch* et F-race ($p = 0.099$),
- *lorsque* $\alpha_f = 0.2$, les trois paires de stratégies deviennent significativement différentes. exploreMaxμ_g surpasse les deux autres stratégies et F-race surpasse *PCSsearch*,
- *lorsque* $\alpha_f = 0.4$, la stratégie *PCSsearch* est significativement inférieure aux deux autres stratégies ($p << 0.05$) mais il n'y a pas de différence statistique suffisante entre les performances de exploreMaxμ_g et F-race ($p = 0.069$).

Dans de nombreuse situations pratiques, les variables associées aux alternatives suivent des lois de distribution normale. Il est à noter que les conditions expérimentales sont ici en faveur d'exploreMaxμ_g et de *PCSsearch*. Ces deux algorithmes font l'hypothèse que les variables associées aux alternatives suivent des lois de distribution normale, ce qui – dans le cas de ces données synthétiques – est vérifié. Le F-race est un algorithme non-paramétrique qui cherche à tirer avantage d'une analyse par bloc, qui n'est ici pas possible. Ces tests ont permis de mesurer les performances de l'algorithme dans le cas gaussien.

5.4.1.3 Sélection de l'optimum lorsque $K = 2$

Dans cette section, les performances de la stratégie optiK2Estim sont comparées aux performances des stratégie $\mathcal{S} + \mathcal{R}$ et F-race dans le cas où le nombre d'alternatives est égal à deux. Les trois problèmes synthétiques utilisés sont décrits dans le tableau 5.4. Comme précédemment, cinq cents tâches sont générées aléatoirement pour chaque problème et les performances

des algorithmes sont comparées à la fin en appliquant des tests statistiques (t-tests jumelés) aux résultats obtenus sur les différentes tâches.

Dans un premier temps, les performances de la stratégie optiK2Estim sont comparées aux performances de la stratégie $\mathcal{S} + \mathcal{R}$ où $\alpha_f = 0.2$. Pour chaque tâche, c'est la stratégie $\mathcal{S} + \mathcal{R}$ qui détermine le nombre de rounds autorisés pour chercher l'optimum et optiK2Estim doit adopter le même nombre d'étapes. Sur les trois problèmes, $\mathcal{S} + \mathcal{R}$ se fait battre trois fois par optiK2Estim mais jamais de manière significative.

Les performances d'optiK2Estim sont ensuite comparées avec celles des trois versions du F-race où le paramètre α_f prend les valeurs 0.05, 0.1 et 0.2. F-race permet de fixer à l'avance le nombre de rounds maximum (H_{max}) autorisés pour la sélection de l'optimum. Ici, le paramètre H_{max} va prendre les valeurs 2000, 1000, 500 et 200 ; ceci permet de multiplier par quatre le nombre de problèmes de sélection, soit un total de douze problèmes.

Dans le cas où α_f vaut 0.05, optiK2Estim surpasse significativement F-race cinq fois sur douze mais l'inverse n'est jamais vrai. Lorsque α_f vaut 0.1, la stratégie optiK2Estim est plus performante que F-race sur deux problèmes et F-race ne surpasse jamais optiK2Estim. Enfin, dans le cas où α_f est égal à 0.2, optiK2Estim est une seule fois significativement supérieur à F-race tandis que F-race n'est jamais significativement supérieur à optiK2Estim.

Dans le cas où $K = 2$, la stratégie optiK2Estim est donc une méthode compétitive avec les stratégies faisant partie de l'état de l'art.

5.4.2 Problèmes de sélection de modèles locaux

Cette section évalue les performances, sur des problèmes de sélection de modèles (voir chapitre 2), des stratégies *PCSsearch* et exploreMaxμ_g comparativement à trois versions de la stratégie F-race où $\alpha_f = \{0.2, 0.1, 0.05\}$.

Nous utilisons, pour réaliser les expériences, vingt-cinq ensembles d'exemples de régression bien connus (voir tableau 5.5). Neuf des vingt-cinq ensembles sont composés d'observations d'entrée/sortie collectées expérimentalement sur des problèmes de régression réelle : abalone [7], ailerons [7], census[7], covtype2 [8], covtype3 [8] elevators [7], housing [7], pol [7] et stock [7]. Les autres ensembles sont générés de manière synthétique. Les huit ensembles de la forme « bank-* » sont créés via un simulateur qui modélise le choix des clients pour une banque [9]. Les huit derniers ensembles de la forme « kin-* » contiennent des données sur la cinématique d'un bras robotisé [10].

Les ensembles d'observations sont répartis cent fois en deux sous-ensembles : un ensemble d'apprentissage de deux cents observations ($\mathbf{D}_N$) et un ensemble de tests utilisé pour valider la sélection. Les trois stratégies doivent

7. http://www.liacc.up.pt/~ltorgo/Regression/DataSets.html
8. http://kdd.ics.uci.edu/databases/covertype/covertype.data.html
9. http://www.cs.toronto.edu/~delve/
10. http://www.ics.uci.edu/~mlearn/MLSummary.html

Name	abalone	ailerons	bank-32fh	bank-32fm	bank-32nh
N	4177	10000	8192	8192	8192
d	10	40	32	32	32
Name	bank-32nm	bank-8fh	bank-8fm	bank-8nh	bank-8nm
N	8192	8192	8192	8192	8192
d	32	8	8	8	8
Name	census	covtype2	covtype3	elevators	housing
N	22784	100000	10000	10000	506
d	137	54	54	18	13
Name	kin32fh	kin32fm	kin32nh	kin32nm	kin8fh
N	8192	8192	8192	8192	8192
d	32	32	32	32	8
Name	kin8fm	kin8nh	kin8nm	pol	stock
N	8192	8192	8192	10000	950
d	8	8	8	48	9

TABLE 5.5 – Les ensembles d'observations utilisés pour comparer les stratégies. N est le nombre d'observations et d symbolise la norme du vecteur d'entrée.

trouver parmi cinq classes d'hypothèses celle qui minimise l'erreur calculée sur l'ensemble de validation. Les classes d'hypothèses sont des KNN (voir section 2.6.2.2 à la page 37) où le nombre de voisins considérés est égal à 2, 5, 10, 15 et 20.

Le nombre de rounds maximum est de $H_{max} = 200$ étapes. F-race pouvant utiliser moins de H_{max} étapes pour choisir l'optimum, c'est donc lui qui fixe le nombre de rounds et les deux autres stratégies doivent en utiliser le même nombre.

Pour chaque ensemble d'apprentissage, les stratégies renvoient le nombre pour le KNN de voisins sélectionnés comme étant l'optimum. Le score d'une stratégie pour ce problème d'apprentissage correspond à l'erreur de validation du KNN utilisant ce nombre de voisins. A la fin des cent problèmes d'apprentissages et pour chacune des stratégies, un vecteur contiendra donc cent mesures de l'erreur du modèle calculées sur l'ensemble de validation. Ces vecteurs seront ensuite utilisés pour comparer (via des t-tests jumelés) les performances des stratégies.

Les résultats expérimentaux sont reportés dans le tableau 5.6. La première partie donne les résultats lorsque les stratégies *PCSsearch* et exploreMaxμ_g sont comparées avec la stratégie F-race où $\alpha_f = 0.05$. Comme précédemment, le chiffre dans la colonne C et la ligne L correspond au nombre de fois que l'algorithme de la ligne L est significativement supérieur à l'algorithme de la colonne C. Pour rappel, il y a un total de vingt-cinq problèmes de régression. Les deux autres parties du tableau donnent les résultats lorsque les deux stratégies *PCSsearch* et exploreMaxμ_g sont comparées avec F-race où le paramètre α_f prend les valeurs 0.1 et 0.2.

Nous observons que les performances de F-race s'améliorent, comparativement aux deux autres stratégies, lorsque α_f augmente. Cela est dû au fait

	PCSsearch	exploreMaxμ_g	F-race ($\alpha_f = 0.05$)
PCSsearch		0	1
exploreMaxμ_g	7		5
F-race ($\alpha_f = 0.05$)	2	0	

	PCSsearch	exploreMaxμ_g	F-race ($\alpha_f = 0.1$)
PCSsearch		0	1
exploreMaxμ_g	9		5
F-race ($\alpha_f = 0.1$)	2	0	

	PCSsearch	exploreMaxμ_g	F-race ($\alpha_f = 0.2$)
PCSsearch		0	1
exploreMaxμ_g	6		2
F-race ($\alpha_f = 0.2$)	4	2	

TABLE 5.6 – Résultats expérimentaux lorsque trois versions de la stratégie F-race sont comparées avec les stratégies *PCSsearch* et exploreMaxμ_g sur *vingt-cinq problèmes* de sélection de modèles.

que H_{max} a été fixé à deux cents et que cette petite valeur n'a pas donné suffisamment de temps à F-race pour supprimer les alternatives les moins compétitives de la course. De manière générale, comparées aux problèmes synthétiques, les performances de F-race sont meilleures. Ceci est dû au fait que l'hypothèse de normalité n'est ici plus vérifiée et que F-race peut utiliser les avantages d'une analyse par bloc.

5.4.3 Discussion

Cette section a comparé les performances des trois algorithmes optiK2-Estim, *PCSsearch* et exploreMaxμ_g avec les performances des algorithmes $\mathcal{S} + \mathcal{R}$ et F-race. Les tests ont été réalisés sur des problèmes générés de manière synthétique et sur des problèmes de sélection de modèles construits sur des données réelles.

De manière générale, les algorithmes $\mathcal{S}+\mathcal{R}$ et *PCSsearch* se sont montrés moins performants :

- Les mauvaises performances de $\mathcal{S} + \mathcal{R}$ sont dues en partie à sa faible capacité d'adaptation. C'est une stratégie en deux étapes d'échantillonnage. Elle n'ajuste qu'une seule fois le nombre de tests qui sera réalisé sur les alternatives. Les deux autres stratégies sont quant à elles séquentielles et peuvent – à chaque étape – s'adapter en choisissant l'alternative à tester.
- La stratégie *PCSsearch* choisit l'alternative à tester en fonction des

valeurs de $\Pr\left\{S_{[K]}^{l+1}(k)\right\}$ avec $k \in [1, K]$. Or, il se trouve que ces différentes valeurs peuvent être très proches l'une de l'autre et comme elles sont estimées, la stratégie peut choisir une mauvaise alternative. Ceci explique en partie les faibles performances de la stratégie. Pour améliorer celles-ci, il faut étudier des méthodes de réduction de la variance d'estimateurs comme par exemple des techniques de combinaison d'estimateurs.

De manière générale, la stratégie exploreMaxμ_g s'est bien comportée en donnant des performances compétitives avec celles des algorithmes de l'état de l'art. Comparées aux performances de F-race, la stratégie exploreMaxμ_g donne de moins bons résultats sur les problèmes réels de sélection de modèles. Ceci est peut être dû au fait qu'exploreMaxμ_g est sensible à la violation de l'hypothèse de normalité. Afin de vérifier cette hypothèse et afin d'améliorer encore davantage les résultats, il pourrait être intéressant d'étudier des méthodes qui ajustent exploreMaxμ_g aux problèmes de sélection de modèles (voir section 4.1.2).

Globalement, la stratégie optiK2Estim a donné de bons résultats comparés aux autres méthodes. Lorsque seules deux alternatives sont en compétition, nous avons vu (section 4.2.1 à la page 77) que pour augmenter la probabilité d'effectuer une sélection correcte ($\Pr\{S_{[K]}\}$), il faut réduire la quantité $\sigma_{k_1}^2/n_{k_1} + \sigma_{k_2}^2/n_{k_2}$. Pour ce faire, il n'est pas utile d'utiliser (et donc d'estimer) les espérances (μ) des deux alternatives. C'est ainsi que procède optiK2Estim qui se base sur l'estimation des deux variances (σ^2) dans le but de tester l'alternative qui réduit le plus $\sigma_{k_1}^2/n_{k_1} + \sigma_{k_2}^2/n_{k_2}$ à l'étape suivante. Ceci explique en partie les bonnes performances d'optiK2Estim comparées aux autres stratégies.

5.5 Conclusions

Une stratégie d'exploration doit, en un budget de tests limité et dans un environnement aléatoire, trouver l'alternative optimale c'est-à-dire celle ayant la plus grande espérance. Ce chapitre propose trois nouveaux algorithmes d'exploration pour la sélection de l'optimum. Nous avons montré que, pour maximiser la probabilité de faire une sélection correcte dans le cas où seules deux alternatives sont en course, il n'est pas utile de connaître leurs espérances. Dans le cas où $K = 2$, les seules variables importantes sont les deux variances et le nombre de fois que les alternatives ont été testées. Le premier algorithme proposé dans ce chapitre est dénoté optiK2Estim. Il est spécifique à cette situation où seules deux alternatives sont en compétition.

Le deuxième algorithme, dénoté *PCSsearch*, propose de remplacer – dans la définition analytique de la probabilité de faire une sélection correcte (voir section 4.1 à la page 70) – les moyennes et les variances par leurs estimations. A chaque étape, l'algorithme sélectionne alors l'alternative qui maximise la

valeur future de cette estimation de la quantité $\Pr\{S_{[K]}\}$.

La section 4.2.2 à la page 84 a étudié l'évolution du gain espéré d'une action gloutonne lorsque les alternatives sont testées. Dans cette section nous avons observé que, dans certaines circonstances, tester une alternative avec une grande espérance pouvait diminuer la quantité μ_g lorsque les variances des alternatives sous-optimales étaient élevées. Le troisième algorithme proposé dans ce chapitre utilise la propriété de l'évolution du gain espéré d'une action gloutonne pour tenter de ne jamais faire diminuer μ_g. Cette stratégie est appelée exploreMaxμ_g.

La section 5.4 compare les trois nouvelles stratégies avec celles faisant partie de l'état de l'art. Les résultats ont montré que ces nouveaux algorithmes sont compétitifs avec les techniques existantes.

Chapitre 6

Algorithmes semi-uniformes pour le problème du *bandit* – contributions

Il existe de nombreux algorithmes pour résoudre le *multi-armed bandit problem* (MABP). Une partie de ceux-ci ont été présentés dans la section 3.3. Comme nous l'avons déjà vu, les stratégies de la famille des stratégies semi-uniformes possèdent deux modes de fonctionnement bien distincts : soit elles explorent les différentes alternatives, soit elles exploitent l'alternative qui semble – pour l'instant – être l'optimum.

Ce chapitre propose deux nouveaux algorithmes semi-uniformes pour résoudre le MABP : ϵ-PCSgreedy et DP-*greedy*.

En mode exploration, un algorithme semi-uniforme sélectionne les alternatives de manière aléatoire selon une distribution de probabilités uniforme. L'algorithme cherche alors à obtenir davantage d'informations sur les alternatives pour que les prochaines actions d'exploitation aient plus de chances de sélectionner une alternative ayant une moyenne élevée. Nous proposons dans ce chapitre de remplacer cette exploration aléatoire uniforme par une exploration qui cherche explicitement à maximiser la probabilité $\Pr\{S_{[K]}\}$ qui est la probabilité qu'une action d'exploitation gloutonne sélectionne l'optimum. Cet algorithme est appelé ϵ-PCSgreedy et utilise *PCSsearch* (voir section 5.2) durant l'exploration.

La majorité des algorithmes semi-uniformes utilisent des stratégies de sélection qui dépendent d'un paramètre (ex. ϵ) généralement fixé empiriquement. Le deuxième algorithme proposé dans ce chapitre est baptisé *Dynamic Programming greedy* (*DP-greedy*). Cet algorithme semi-uniforme combine des techniques d'estimation avec des méthodes de programmation dynamique (voir annexe D) pour ajuster automatiquement l'exploration et l'exploitation. L'idée est d'estimer, à partir des données disponibles, les quantités μ_g et μ_r qui sont respectivement les gains espérés d'une action d'exploi-

tation gloutonne (voir chapitre 4) et d'une action d'exploration aléatoire uniforme. Ces deux estimations sont ensuite utilisées au sein d'un programme dynamique pour donner un « *score* » aux actions gloutonne et aléatoire et c'est alors l'action obtenant le plus grand score qui est exécutée. Comme nous le verrons, cette manière de procéder permet à la stratégie d'adopter un comportement adaptatif en fonction du problème. Lorsque le problème est difficile ou lorsque peu d'observations ont été collectées, la stratégie se trouve automatiquement dans un mode d'exploration forte et, au fur et à mesure que les observations sont collectées, DP-*greedy* bascule dans un mode de plus forte exploitation.

Ce chapitre contient les contributions de cette thèse en rapport avec le MABP. La section 6.1 reformule le MABP lorsque l'information sur les différentes alternatives est parfaite ou imparfaite. Les deux algorithmes proposés dans cette section sont d'abord définis lorsque l'information est parfaite et des méthodes d'estimation sont ensuite utilisées pour adapter les algorithmes lorsque l'information est imparfaite. La section 6.2 présente ϵ-PCSgreedy et la section 6.3 présente DP-*greedy*. La dernière section 6.4 contient les expériences réalisées sur des problèmes créés de manière synthétique et sur des problèmes élaborés à partir de données réelles.

6.1 Reformulation du problème lorsque l'information sur l'état du système est parfaite ou imparfaite

Le MABP a été modélisé en section 3.3.1 par un ensemble $\{\mathbf{z}_k\}_{k\in[1,K]}$ de K variables aléatoires $\mathbf{z}_k$ de moyenne μ_k et d'écart-type σ_k où la meilleure alternative a l'indice $[K] = \arg\max \mu_k$.

Le problème est divisé en H étapes. A chaque étape l, le joueur doit choisir le bras qui sera testé. La quantité $\mathbf{n}_k$ est le nombre de fois que le bras k a été testé durant les l premiers rounds et $\mathbf{Z}_k(\mathbf{n}_k) = \left[\mathbf{z}_k^1, \mathbf{z}_k^2, \ldots, \mathbf{z}_k^{\mathbf{n}_k}\right]$ est un ensemble qui contient $\mathbf{n}_k$ variables aléatoires indépendantes et identiquement distribuées.

Le MABP est un exemple de problème où l'information sur l'état du système (les caractéristiques de la machine à sous) est imparfaite (en anglais *Imperfect State Information* (ISI) [12]) c'est-à-dire que le processus stochastique sous-jacent qui définit les gains que reçoit le joueur lorsqu'il décide de tester une alternative lui est inconnu et que seule une série de quelques observations des gains est disponible. L'état du système doit être estimé sur ces quelques observations.

Ce chapitre propose deux nouveaux algorithmes pour résoudre le MABP. L'approche que nous adoptons est identique à celle utilisée pour résoudre certains problèmes de programmation dynamique dans les cas ISI [12]. Il s'agit d'abord de résoudre le problème lorsque l'information sur l'état du

système est parfaite (en anglais *Perfect State Information* (PSI) [12]) et ensuite d'utiliser des méthodes d'estimation sur les variables inconnues. Il s'agit d'une approche classique de résolution des problèmes de programmation dynamique lorsque le problème est linéaire et la fonction de coût quadratique. Cette approche est appelée *certainty equivalence*.

Dans la configuration PSI, on définit (i) l'état $s^l \in \mathcal{S}$ du système à l'étape l par l'ensemble $\langle \{\mu_k\}, \{\sigma_k\}, \{n_k^l\} \rangle_{k=1,\dots,K}$ et (ii) la stratégie du joueur comme une fonction $\pi : \mathcal{S} \to \{1, \dots, K\}$ qui renvoie pour chaque état s le bras qui devra être testé.

Dans la configuration ISI, on définit (i) l'état $\widehat{\mathbf{s}}^l \in \widehat{\mathcal{S}}$ du système à l'étape l par l'ensemble de toutes les observations collectées $\{\mathbf{Z}_k(n_k^l)\}_{k=1,\dots,K}$ et (ii) la stratégie du joueur comme une fonction $\widehat{\pi} : \widehat{\mathcal{S}} \to \{1, \dots, K\}$ qui renvoie pour chaque état $\widehat{\mathbf{s}}$ le bras $\widehat{\mathbf{k}}$ qui devra être testé.

Deux stratégies habituellement utilisées dans le cas ISI sont les stratégies aléatoire et gloutonne. La stratégie aléatoire néglige toute l'information collectée sur l'état du système à l'étape l et sélectionne

$$\widehat{\pi}(\widehat{\mathbf{s}}^l) = \widehat{\mathbf{k}} \sim \text{Uni}(1/K, \dots, 1/K)$$

l'alternative à tester aléatoirement selon une distribution uniforme.

La stratégie gloutonne, quant à elle, utilise l'information partielle contenue dans $\widehat{\mathbf{s}}$ et renvoie

$$\widehat{\mathbf{k}} = [\widehat{\mathbf{K}}] = \arg\max \widehat{\boldsymbol{\mu}}_k^l$$

où $\widehat{\boldsymbol{\mu}}_k^l$ est la moyenne arithmétique des gains de la k-ième alternative à l'étape l.

Le vecteur $\mathbf{Z}_k(n_k^l)$ est un vecteur aléatoire et donc l'état du système $\widehat{\mathbf{s}}^l$ et la sortie de la fonction $\widehat{\pi}(\widehat{\mathbf{s}})$ sont également aléatoires. Comme nous l'avons déjà vu à la section 3.3.1, il est possible de définir le regret à l'étape l d'une stratégie $\widehat{\pi}$ par

$$\delta_{\widehat{\pi}} = \mu_{[K]} - \mathbb{E}\big[\mathbf{z}_{\widehat{\mathbf{k}}}\big] \tag{6.1}$$

qui quantifie la perte des gains suite à l'adoption de la stratégie à l'étape l.

Nous allons maintenant définir le regret à l'étape l d'une stratégie aléatoire uniforme et d'une stratégie gloutonne.

Le regret d'une stratégie de sélection aléatoire est

$$\begin{aligned}\delta_r &= \mu_{[K]} - \frac{1}{K}\sum_{k=1}^{K}\mu_k \\ &= \mu_{[K]} - \mu_r\end{aligned}$$

où le terme

$$\mu_r = \frac{1}{K}\sum_{k=1}^{K}\mu_k$$

désigne le *gain espéré d'une action d'exploration aléatoire*.

Le regret d'une stratégie de sélection gloutonne est

$$\begin{aligned}\delta_g &= \mu_{[K]} - \sum_{k=1}^{K} \Pr\{S_k\} \cdot \mu_k \\ &= \mu_{[K]} - \mu_g\end{aligned}$$

où

$$\Pr\{S_k\} = \Pr\left\{k = \arg\max_i \{\widehat{\boldsymbol{\mu}}_i\}\right\} \tag{6.2}$$

correspond à la probabilité que l'alternative k soit sélectionnée par un algorithme glouton (section 4.1) et où

$$\mu_g = \sum_{k=1}^{K} \Pr\{S_k\} \cdot \mu_k \tag{6.3}$$

désigne le *gain espéré d'une action d'exploitation gloutonne*. La probabilité de faire une sélection correcte avec une action gloutonne (en anglais *Probability of Correct Selection* (PCS)) est $\Pr\{S_{[K]}\}$. Une définition analytique de $\Pr\{S_k\}$ a été donnée dans la section 4.1 à la page 70. Dans le cas PSI, cette quantité dépend des espérances des gains $\{\mu_k\}$, des écarts-types des distributions des gains $\{\sigma_k\}$ et du nombre de tests déjà réalisés sur les différents bras $\{n_k\}$.

6.2 La stratégie ϵ-PCSgreedy

Cette section présente la stratégie ϵ-PCSgreedy. Lorsqu'une stratégie semi-uniforme est en mode d'exploration, elle sélectionne généralement l'alternative à tester de manière aléatoire selon une distribution de probabilités uniforme.

Le rôle d'une exploration est de maximiser les gains des prochaines actions d'exploitation. Pour cela, les quantités $\Pr\{S_{[K]}\}$ et μ_g sont de bonnes mesures de l'efficacité espérée des prochaines actions gloutonnes. Si le nombre de tests sur les K alternatives tend vers l'infini alors $\Pr\{S_{[K]}\}$ converge vers un et par voie de conséquence μ_g converge vers $\mu_{[K]}$ (voir section 4.2). Une exploration aléatoire uniforme garantit donc bien qu'asymptotiquement, les sélections gloutonnes ne sélectionnent plus que l'optimum $[K]$.

Dans cette section, nous proposons d'améliorer la phase d'exploration des algorithmes semi-uniformes ϵ-greedy en remplaçant la sélection aléatoire par une sélection basée sur la probabilité de faire un choix correct. Ceci a pour but d'accélérer l'augmentation de la valeur $\Pr\{S_{[K]}\}$ vers un. Lors de l'exploration, ϵ-PCSgreedy va utiliser l'algorithme *PCSsearch* déjà décrit dans la section 5.2.

Algorithme 17 La stratégie ϵ-PCSgreedy

Tester au minimum deux fois chaque alternative.
boucler
  $\mathbf{e} \leftarrow U[0,1]$ #distribution uniforme
  si $\mathbf{e} < \epsilon$ **alors**
    pour $k = 1$ à K **faire**
      $n_{\#}^{l+1} \leftarrow \left\{n_1^l, \ldots, n_k^l + 1, \ldots, n_K^l\right\}$.
      $\mathbf{PCS}_k^{l+1} \leftarrow calculerPCS\left(\{\widehat{\boldsymbol{\mu}}_k\}_{k\in\{1,K\}}, \{\widehat{\boldsymbol{\sigma}}_k\}_{k\in\{1,K\}}, n_{\#}^{l+1}\right)$, .
    fin pour
    $\widehat{\mathbf{k}} \leftarrow \arg\max_k \left\{\mathbf{PCS}_k^{l+1}\right\}$.
  sinon
    $\widehat{\mathbf{k}} \leftarrow \arg\max_k \{\widehat{\boldsymbol{\mu}}_k\}$.
  fin si
  Tester l'alternative $\widehat{\mathbf{k}}$.
fin boucle

Cette stratégie (voir algorithme 17) correspond à la stratégie *ϵ-greedy* (voir algorithme 9 à la page 62) où l'étape de sélection aléatoire est remplacée par l'algorithme *PCSsearch*. Dans la section 6.4.1, cette stratégie sera testée expérimentalement parmi d'autres stratégies semi-uniformes déjà décrites dans la section 3.3.2.

D'un point de vue computationnel, c'est l'appel à la fonction $calculerPCS$ qui demande le plus grand effort de calcul. Pour un problème à horizon H avec K alternatives cette fonction est exécutée en moyenne $\epsilon \times H \times K$ fois. Une attention particulière doit donc être apportée à l'optimisation de la vitesse d'exécution de cette fonction.

6.3 La stratégie DP-*greedy*

Cette section introduit un autre nouvel algorithme dans la famille des algorithmes semi-uniformes, appelé *Dynamic Programming greedy* (*DP-greedy*). Les algorithmes semi-uniformes sont caractérisés par l'alternance de deux modes de fonctionnement : un mode d'exploitation et un mode d'exploration. Selon la définition du gain espéré d'une action d'exploitation gloutonne (voir équation (6.3)), la performance d'une stratégie gloutonne à l'étape l dépend des probabilités $\Pr\{S_k\}$. En particulier, plus la probabilité $\Pr\left\{S_{[K]}\right\}$ de faire une sélection correcte avec une action gloutonne est proche de un, plus le gain d'une action gloutonne a des chances d'être proche de $\mu_{[K]}$.

En début de jeu, c'est-à-dire lorsque l est petit, la probabilité $\Pr\left\{S_{[K]}\right\}$ de faire une sélection correcte avec une action d'exploitation gloutonne est généralement faible et il est alors préférable de poser des actions d'exploration dans le but de faire converger la probabilité $\Pr\left\{S_{[K]}\right\}$ vers un. Ces

considérations qualitatives, communes à tous les problèmes d'exploration / exploitation [82], peuvent être définies formellement en décrivant comment, à chaque étape du MABP, les gains moyens des actions aléatoire et gloutonne évoluent. Dans ce but, il est intéressant de reformuler le problème en utilisant les processus de décision de Markov [69] à horizon fini et d'étudier la solution optimale renvoyée par un algorithme de programmation dynamique [12].

Le MABP reflète une situation où les décisions doivent être prises dans un environnement où l'information sur le système sous-jacent est incertaine (ISI). En programmation dynamique, une solution typique pour résoudre ce type de problème consiste à le décomposer en deux problèmes indépendants : un problème d'estimation et un problème de contrôle dans le cas où l'information sur l'état du système est parfaite (PSI) [12]. C'est cette approche qui sera appliquée pour définir notre algorithme DP-*greedy*. L'annexe D rappelle les notions de programmation dynamique indispensables à la bonne compréhension de la suite de ce chapitre.

6.3.1 L'algorithme semi-uniforme DP-*greedy* dans le cas PSI

L'algorithme DP-*greedy* considère le MABP comme un problème de programmation dynamique à horizon fini de V étapes [1] dans lesquelles, à chaque étape, seules deux actions sont disponibles dans l'ensemble des actions U : une action d'exploration aléatoire uniforme r et une action d'exploitation gloutonne g. Pour DP-*greedy*, une stratégie $\widetilde{\pi}$ est une fonction $\widetilde{\pi} : \mathcal{S} \rightarrow \{r, g\}$ qui renvoie pour chaque état s une action (soit exploration aléatoire soit exploitation gloutonne).

Il faut noter que toutes les stratégies présentées jusqu'à présent dans cette thèse étaient des fonctions qui renvoyaient le numéro du bras $\{1, \ldots, K\}$ qu'il faut tester. Ici, la stratégie DP-*greedy* prend ses décisions dans un autre espace d'action de contrôle et plutôt que de choisir directement l'alternative à tester, DP-*greedy* détermine le type d'action à poser (exploration ou exploitation).

Nous définissons $s^v = \langle \{\mu_k\}, \{\sigma_k\}, \{n_k^v\} \rangle$, $k = 1, \ldots, K$ comme l'état du système à l'étape v. Une transition depuis l'état s^v où l'alternative k est testée a pour conséquence que n_k^v est incrémenté de un. Nous définissons $s_k^{v+1} = \langle \{\mu_k\}, \{\sigma_k\}, \{n_1^v, \ldots, n_k^v + 1, \ldots, n_K^v\} \rangle$ comme l'état successeur de l'état s^v lorsque l'alternative k est testée. Selon l'équation (6.2) à la page 124, $\Pr\{S_k\}$ dénote la probabilité d'avoir une transition de l'état s^v vers l'état s_k^{v+1} si une action d'exploitation gloutonne est exécutée tandis que $1/K$ est la probabilité de transition de s^v vers s_k^{v+1} si une action d'exploration aléatoire uniforme est exécutée. Il est à noter également que quelle que soit

1. V est le nombre d'étapes – dans le futur – considéré par le programme dynamique. Si H est le nombre de rounds total et que l est le nombre de rounds déjà effectué alors V à une valeur de $H - l$.

l'action posée, une transition de l'état s^v vers l'état s_k^{v+1} renvoie un gain moyen de μ_k.

Une fois le problème interprété comme un problème de programmation dynamique, il est possible de définir l'équation récursive de Bellman associée au problème (voir annexe D). La somme des gains maximum pouvant être obtenue sur V étapes si l'état initial vaut s^v, est

$$J_V^* (s^v) = \max \left[A_g^V (s^v) \ , \ A_r^V (s^v) \right] \tag{6.4}$$

où

$$\begin{aligned} A_g^V (s^v) &= \sum_{k=1}^{K} \Pr\{S_k \mid s^v\} \cdot \left(\mu_k + J_{V-1}^* \left(s_k^{v+1}\right)\right) \\ &= \mu_g (s^v) + \sum_{k=1}^{K} \Pr\{S_k \mid s^v\} \cdot J_{V-1}^* \left(s_k^{v+1}\right) \end{aligned} \tag{6.5}$$

et où

$$\begin{aligned} A_r^V (s^v) &= \sum_{k=1}^{K} \frac{1}{K} \cdot \left(\mu_k + J_{V-1}^* \left(s_k^{v+1}\right)\right) \\ &= \mu_r + \sum_{k=1}^{K} \frac{1}{K} \cdot J_{V-1}^* \left(s_k^{v+1}\right). \end{aligned} \tag{6.6}$$

Dans l'équation (6.5), la quantité $\Pr\{S_k \mid s^v\}$ est la probabilité de sélectionner l'alternative k au moyen d'une action gloutonne si l'état est s^v.

Supposons être dans l'état s^v, la quantité $A_g^V (s^v)$ est le gain maximum pouvant être obtenu sur V étapes si – à cette étape – c'est une action gloutonne qui est exécutée. $A_g^V (s^v)$ est composé de deux termes (voir équation 6.5). Le premier terme $\mu_g (s^v)$ correspond au gain moyen immédiat obtenu par le fait que le joueur décide d'effectuer une action d'exploitation. Le deuxième terme

$$\sum_{k=1}^{K} \Pr\{S_k \mid s^v\} \cdot J_{V-1}^* \left(s_k^{v+1}\right)$$

concerne les gains futurs espérés du joueur après avoir exécuté une action gloutonne. Pour calculer ces gains futurs espérés, on utilise la probabilité $\Pr\{S_k\}$ que – suite à une action gloutonne – l'alternative k soit testée et cette probabilité est multipliée par le gain maximum pouvant être obtenu sur $V-1$ étapes si l'état est s_k^{v+1}.

L'explication pour la quantité $A_r^V (s^v)$ est la même excepté que la probabilité de tester l'alternative k suite à une action de sélection aléatoire uniforme est ici de $1/K$. Finalement, dans l'équation (6.4), le gain maximum pouvant être obtenu sur V étapes est le maximum entre $A_g^V (s^v)$ et $A_r^V (s^v)$.

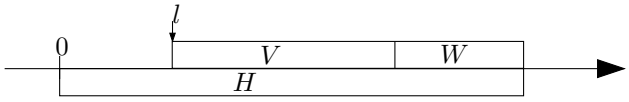


FIGURE 6.1 – Notations. L'horizon H est le nombre total de rounds. L'étape l est le nombre de rounds que le joueur a déjà joués. V est le nombre d'étapes de l'algorithme de programmation dynamique. W est le nombre d'actions de pure exploitation gloutonne utilisées pour calculer le gain final $J_0^* (s^v)$.

Selon l'équation (6.4), une stratégie semi-uniforme optimale dans l'état s^v doit exécuter l'action

$$u^* = \begin{cases} g & \text{si } A_g^V (s^v) - A_r^V (s^v) > 0 \\ r & \text{si } A_g^V (s^v) - A_r^V (s^v) < 0. \end{cases}$$

Malgré le caractère optimal de l'algorithme, cette méthode – comme tout programme dynamique – ne peut pas être utilisée si l'horizon est trop grand car la puissance de calcul nécessaire devient vite trop élevée [68]. Dans ce cas, une solution possible consiste à (i) réduire le nombre de V étapes à une valeur plus petite que $H - l$ (par exemple $H - l - W$) et (ii) définir le gain du dernier état $J_0^* (s^v)$ égal au gain espéré μ_G d'une série de W actions de pure exploitation gloutonne où $W = H - l - V$ (voir figure 6.1). Pour un état s^w, la quantité μ_G est récursivement définie comme suit

$$\begin{aligned} \mu_G^W (s^w) &= \sum_{k=1}^{K} \Pr\{S_k \mid s^w\} \cdot \left(\mu_k + \beta \cdot \mu_G^{W-1} \left(s_k^{w+1} \right) \right) \qquad (6.7) \\ \mu_G^1 (s^w) &= \mu_g (s^w), \end{aligned}$$

où $\Pr\{S_k \mid s^w\}$ est la probabilité d'aller de l'état s^w vers l'état $s_k^{w+1} = \langle \{\mu_k\}, \{\sigma_k\}, \{n_1^w, \ldots, n_k^w + 1, \ldots, n_K^w\} \rangle$ après une action gloutonne. La quantité β $(0 < \beta \leq 1)$ est un facteur de réduction – défini par l'utilisateur – qui a pour conséquence de donner plus d'importance aux gains à court terme.

Cette section a montré que la quantité μ_g (voir équation (6.3)) et la quantité associée $\Pr\{S_k\}$ (voir équation (6.2)) doivent être connues si l'on veut implémenter l'algorithme semi-uniforme DP-*greedy*. Malheureusement, ces quantités ne sont en réalité pas accessibles au joueur et une procédure d'estimation est requise si l'on veut utiliser l'algorithme en pratique. Il est néanmoins intéressant de remarquer que la définition d'une stratégie semi-uniforme optimale repose sur deux quantités qui sont en rapport avec la performance d'une stratégie de pure exploitation gloutonne et d'une stratégie d'exploration aléatoire.

Comme dans la stratégie d'indice de Gittins (voir section 3.3.5 à la page 66), notre méthode est construite dans un cadre de programmation

dynamique. Néanmoins, dans la formulation du MABP de Gittins, les variables dans l'ensemble des actions possibles sont les différents bras. Gittins donne un score à chacun et le bras présentant le plus haut score est joué. Dans notre approche, les actions ne sont pas directement posées au niveau des bras mais plutôt au niveau de la stratégie. L'ensemble des actions possibles contient ici deux valeurs : soit faire de l'exploration aléatoire (r) soit faire de l'exploitation gloutonne (g). DP-*greedy* donne un score aux deux actions et l'action présentant le plus grand score est exécutée.

La section suivante utilise les méthodes d'estimation présentées dans la section 4.4 à la page 94 pour proposer une version de l'algorithme semi-uniforme DP-*greedy* dans le cas ISI.

6.3.2 L'algorithme semi-uniforme DP-*greedy* dans le cas ISI

La section 6.3.1 décrit la version PSI de DP-*greedy* où la stratégie prend avantage d'une parfaite connaissance du système pour calculer les valeurs exactes de A_g et de A_r. C'est une hypothèse bien entendu irréaliste. Cette section présente la version ISI de DP-*greedy* où, à chaque étape, DP-*greedy* est supposé n'avoir qu'une information partielle et bruitée de l'état du système.

Dans cette version de l'algorithme DP-*greedy*, durant une phase d'initialisation, chaque bras est testé I fois, ce qui permet d'obtenir une distribution a priori de l'état du système. Supposons qu'à l'étape l, le bras k soit testé et que l'on obtienne un gain de $\mathbf{z}_k^{n_k^l+1}$. Ce gain est ajouté au vecteur $\mathbf{Z}_k(n_k^l) = [\mathbf{z}_k^1,\ \mathbf{z}_k^2,\ \ldots,\ \mathbf{z}_k^{n_k^l}]$ des observations collectées sur l'alternative k et $\widehat{\mathbf{s}}^l = \left\{\mathbf{Z}_k(n_k^l)\right\}_{k=1,\ldots,K}$ est l'état du programme dynamique dans le cas où l'information sur le système est incomplète.

Dans le but de réduire la complexité de calcul de DP-*greedy* présenté dans la section 6.3.1, on utilise une version de la stratégie où $V = 1$ ce qui donne [2]

$$A_g\left(s^v\right) = \sum_{k=1}^{K} \Pr\{S_k \mid s^v\} \cdot \left(\mu_k + \mu_G^W\left(s_k^{v+1}\right)\right) \tag{6.8}$$

et

$$A_r\left(s^v\right) = \frac{1}{K}\sum_{k=1}^{K}\left(\mu_k + \mu_G^W\left(s_k^{v+1}\right)\right). \tag{6.9}$$

Les techniques d'estimation présentées dans la section 4.4 peuvent maintenant être utilisées pour estimer A_g et A_r.

2. A titre d'indication, nous avons également réalisé de petits tests dans le cas où $V = 2$. Les résultats ne se sont pas révélés différents mais d'un point de vue computationnelle les temps de calcul se sont révélés excessifs. C'est la raison pour laquelle nous nous sommes limités au cas $V = 1$.

C'est la technique d'estimation par *mise de côté* qui est utilisée car celle-ci permet de calculer une estimation de μ_g pour d'autres valeurs de $\{n_k\}$ (voir section 4.4.3).

Pour A_g, l'estimation de la probabilité $\Pr\{S_k \mid s^v\}$ va être décorrélée des estimations de μ_k et de $\mu_G^W\left(s_k^{v+1}\right)$. Pour cela, les observations dans $\left\{\mathbf{Z}_k(n_k^l)\right\}_{k=1,\ldots,K}$ sont partitionnées en deux parties pour calculer $\widehat{\boldsymbol{\mu}}_k^A$, $\widehat{\boldsymbol{\sigma}}_k^A$, $\widehat{\boldsymbol{\mu}}_k^B$ et $\widehat{\boldsymbol{\sigma}}_k^B$ (voir les équations (4.11), (4.12), (4.13) et (4.14) à la page 96) et ensuite les estimations *plug-in* de $\Pr\{S_k\}$ sont calculées ; c'est-à-dire $\widehat{\mathbf{Pr}}\left\{S_k\right\}^A$ et $\widehat{\mathbf{Pr}}\left\{S_k\right\}^B$. Nous calculons également $\left(\widehat{\boldsymbol{\mu}}_g^{SPL2}\right)^A$ et $\left(\widehat{\boldsymbol{\mu}}_g^{SPL2}\right)^B$ qui sont des estimations de μ_g calculées avec la première partie et la deuxième partie des observations en utilisant la méthode de *mise de côté* (voir section 4.4.3 à la page 96). Nous allons utiliser une version plus robuste de l'estimateur de A_g en calculant deux fois cette quantité sur les deux moitiés de $\left\{\mathbf{Z}_k(n_k^l)\right\}_{k=1,\ldots,K}$ et l'estimateur final consistera en la moyenne des deux estimateurs. L'estimation de A_g est alors

$$\begin{aligned}\widehat{\mathbf{A}}_g = \frac{1}{2}\sum_{k=1}^{K}\widehat{\mathbf{Pr}}\left\{S_k\right\}^A \cdot \left(\widehat{\boldsymbol{\mu}}_k^B + \sum_{i=0}^{W-1}\beta^i \cdot \left(\widehat{\boldsymbol{\mu}}_g^{SPL2}\right)^B\right) \\ + \frac{1}{2}\widehat{\mathbf{Pr}}\left\{S_k\right\}^B \cdot \left(\widehat{\boldsymbol{\mu}}_k^A + \sum_{i=0}^{W-1}\beta^i \cdot \left(\widehat{\boldsymbol{\mu}}_g^{SPL2}\right)^A\right)\end{aligned}$$

où $\sum_{i=0}^{W-1}\beta^i \cdot \mu_g\left(s_k^{v+1}\right)$ est une approximation de $\mu_G^W\left(s_k^{v+1}\right)$.

Pour estimer A_r, il n'est pas nécessaire de décorréler $1/K$ avec les estimations de μ_k et de $\mu_G^W\left(s_k^{v+1}\right)$. Les estimations $\widehat{\boldsymbol{\mu}}_k$ et $\widehat{\boldsymbol{\mu}}_g^{SPL2}$ des quantités μ_k et μ_g sont d'abord calculées avec toutes les observations de l'ensemble $\left\{\mathbf{Z}_k(n_k^l)\right\}_{k=1,\ldots,K}$ et l'estimation de A_r est alors

$$\widehat{\mathbf{A}}_r = \frac{1}{K}\sum_{k=1}^{K}\left(\widehat{\boldsymbol{\mu}}_k + \sum_{i=0}^{W-1}\beta^i \cdot \widehat{\boldsymbol{\mu}}_g^{SPL2}\right).$$

La stratégie DP-*greedy* dans le cas ISI consiste dans ce cas à effectuer l'action

$$\mathbf{u}^* = \begin{cases} g & \text{si } \widehat{\mathbf{A}}_g - \widehat{\mathbf{A}}_r > 0 \\ r & \text{si } \widehat{\mathbf{A}}_g - \widehat{\mathbf{A}}_r < 0 \end{cases}.$$

L'algorithme 18 donne le pseudo-code de DP-*greedy*. Durant l'initialisation, chaque alternative est testée I fois (ligne 1). Ensuite, pour chaque étape, DP-*greedy* calcule $\widehat{\mathbf{A}}_g$ et $\widehat{\mathbf{A}}_r$ (ligne 3) et c'est en fonction de ces deux valeurs que DP-*greedy* choisit l'action à exécuter. Si $\widehat{\mathbf{A}}_g - \widehat{\mathbf{A}}_r > 0$ alors c'est une action d'exploitation gloutonne qui est exécutée, dans le cas contraire il s'agira d'une action d'exploration aléatoire.

Dans la section 6.4.2, cette stratégie sera testée expérimentalement parmi d'autres stratégies semi-uniformes déjà décrites dans la section 3.3.2.

Algorithme 18 L'algorithme DP-*greedy*

tester chaque bras I fois
boucler
  Calculer les valeurs de $\widehat{\mathbf{A}}_g$ et $\widehat{\mathbf{A}}_r$.
  si $\widehat{\mathbf{A}}_g - \widehat{\mathbf{A}}_r > 0$ **alors**
    $\widehat{\mathbf{k}} \leftarrow \arg\max_k \{\widehat{\boldsymbol{\mu}}_k\}$
  sinon
    $\widehat{\mathbf{k}} \leftarrow$ aléatoire
  fin si
  Tester $\mathbf{z}_{\widehat{\mathbf{k}}}$
fin boucle

6.4 Expérimentations et discussion

Cette section est divisée en deux parties. La première partie évalue les performances de l'algorithme ϵ-PCSgreedy décrites dans la section 6.2 et la deuxième partie se penche sur les performances de la version ISI de l'algorithme DP-*greedy* détaillées dans la section 6.3.2.

6.4.1 Expériences sur la stratégie ϵ-PCSgreedy

Cette section évalue les performances de la stratégie ϵ-PCSgreedy en la comparant à d'autres stratégies semi-uniformes présentées dans la section 3.3.2. Nous allons montrer que remplacer l'exploration aléatoire uniforme dans la stratégie ϵ-*greedy* par *PCSsearch* améliore les performances de cette stratégie. Quatre stratégies de *bandit* sont testées : deux instances de ϵ-*greedy* ($\epsilon = 0.05$ et $\epsilon = 0.10$) et deux instances de ϵ-PCSgreedy ($\epsilon = 0.05$ et $\epsilon = 0.10$). Quatre problèmes de *bandit* vont être utilisés : les deux premiers problèmes sont créés de manière synthétique et les deux autres sont élaborés à partir de données réelles. C'est le regret cumulé (voir équation (3.11) à la page 59) qui est utilisé pour comparer les performances des stratégies.

Les deux problèmes synthétiques ont un horizon de $H = 1000$ rounds et sont appelés P1 et P2. Les problèmes se distinguent par le nombre d'alternatives : $K = 10$ pour P1 et $K = 15$ pour P2. Chaque problème synthétique est constitué de mille tâches générées de manière aléatoire. Ces tâches sont obtenues en échantillonnant de manière uniforme les moyennes ($\mu_k = U[0, 1]$) et les écarts-types ($\sigma_k = U[0, 2]$) des alternatives. Les performances des stratégies sont obtenues en moyennant le regret cumulé sur les différentes tâches.

Pour les deux problèmes générés à partir de données réelles, nous reprenons les mêmes données que celles utilisées dans [86]. Dans ces problèmes, un programme informatique doit récupérer sur Internet des informations à partir de différentes sources. A chaque étape, le programme choisit une

Problème P1				
H	0.05-greedy	0.05-PCSgreedy	0.10-greedy	0.10-PCSgreedy
200	21.50	21.28	24.01	23.71
400	39.49	38.61	44.1	42.75
600	56.06	54.59	62.21	59.47
800	71.22	69.37	78.93	74.71
1000	85.20	83.16	94.59	89.14
moyenne	54.69	53.40	60.77	57.96

Problème P2				
H	0.05-greedy	0.05-PCSgreedy	0.10-greedy	0.10-PCSgreedy
200	21.81	21.78	24.95	24.90
400	38.97	38.75	45.61	44.44
600	54.53	54.01	64.46	62.15
800	69.31	68.25	82.16	78.38
1000	83.47	81.64	98.9	93.53
moyenne	53.62	52.89	63.22	60.68

TABLE 6.1 – Pour les deux problèmes synthétiques, la première colonne indique l'horizon H et les autres colonnes contiennent le regret cumulé des quatre méthodes. La dernière ligne – appelée moyenne – donne le regret de la méthode moyennée sur tout l'horizon.

source et attend que l'information lui soit transmise. Le but du programme est de minimiser, durant les étapes successives, le temps d'attente. Pour simuler les différents temps d'attente, une base de données a été construite en interrogeant les pages d'accueil de 700 universités (toutes les dix minutes durant plus ou moins dix jours) et en enregistrant chaque fois le temps d'attente (en millisecondes) pour recevoir la page d'accueil [3]. Si l'on interprète cette tâche comme un problème de *bandit*, les pages d'accueil des universités représentent les bras et les temps d'attente correspondent aux gains (négatifs) si une alternative est testée. Dans nos expériences, afin de générer un grand nombre de tâches, nous avons sélectionné mille fois $K = 10$ et $K = 20$ universités et calculé les performances des stratégies sur un horizon de 500 étapes. Ces deux problèmes sont dénotés P3 et P4.

Les tableaux 6.1 et 6.2 donnent le regret (à minimiser) obtenu par les stratégies (.05-greedy, .05-PCSgreedy, .1-greedy et .1-PCSgreedy) sur les quatre problèmes (P1, P2, P3 et P4). Un t-test jumelé est utilisé pour comparer le regret de la méthode ϵ-*greedy* avec celui de la méthode ϵ-PCSgreedy correspondant (ayant la même valeur ϵ). Une valeur soulignée signifie que le regret est significativement ($p < 0.01$) moins élevé que le regret de la stratégie correspondante avec la même valeur de ϵ.

Les résultats montrent que ϵ-PCSgreedy surpasse significativement ϵ-greedy neuf fois sur vingt sur les problèmes synthétiques et douze fois sur vingt sur les problèmes construits à partir des données réelles. A aucun moment, ϵ-*greedy* ne surpasse significativement ϵ-PCSgreedy. Comme attendu,

3. Les données peuvent être téléchargées depuis `http://sourceforge.net/projects/bandit/`

Problème P3				
H	ϵ-greedy.05	PCSgreedy.05	ϵ-greedy.1	PCSgreedy.1
100	7080	6781	9147	8298
200	15310	14424	20198	17852
300	22978	23124	31426	28289
400	30914	30642	42963	38300
500	38583	37220	53658	46551
moyenne	22973	22438	31479	27858

Problème P4				
H	ϵ-greedy.05	PCSgreedy.05	ϵ-greedy.1	PCSgreedy.1
100	5516	4763	7032	5977
200	12650	11107	17168	14806
300	20055	18472	27866	24485
400	27394	25832	38798	34019
500	34551	33280	49438	43300
moyenne	20033	18691	28060	24518

TABLE 6.2 – Pour les deux problèmes basés sur des données réelles, la première colonne indique l'horizon H et les autres colonnes contiennent le regret cumulé des quatre méthodes. La dernière ligne – appelée moyenne – donne le regret de la méthode moyennée sur tout l'horizon.

	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12
K	3	5	10	3	5	10	3	5	10	3	5	10
σ_k	0.1	0.1	0.1	1	1	1	2	2	2	3	3	3

TABLE 6.3 – Les douze cas-tests synthétiques se distinguent par le nombre de bras K et l'écart-type des gains σ_k.

l'amélioration est plus marquée lorsque ϵ est plus grand puisque dans ce cas la méthode d'exploration améliorée est appelée plus souvent et a donc plus de chances de montrer sa valeur ajoutée.

6.4.2 Expériences sur la stratégie DP-*greedy*

Cette section présente les expériences qui ont été réalisées pour évaluer les performances de DP-*greedy* et qui ont été comparées à d'autres stratégies semi-uniformes faisant partie de l'état de l'art. Dix-neuf stratégies semi-uniformes sont testées : dix stratégies ϵ-*greedy* avec $\epsilon = \{0.00, 0.05, 0.10, \ldots, 0.45\}$, huit stratégies ϵ_l-greedy (noté ϵ_0-Dgreedy) avec $\epsilon_0 = \{1, 20, 40, 60, 80, 120, 160, 200\}$ et notre algorithme DP-*greedy* où le terme β est fixé à 0.98.

Un ensemble de quinze cas-tests, dénotés respectivement : B1, B2,..., B15, est utilisé. Les douze premiers cas-tests sont générés de manière synthétique et les trois derniers à partir de données réelles. Chacune des stratégies est initialisée à partir d'un ensemble de $I = 6$ observations.

En ce qui concerne les cas-tests générés de manière synthétique, l'horizon est fixé à $H = 4000$ rounds. Chaque cas-test synthétique est composé de cent tâches de *bandit* générées de manière aléatoire. Ils sont obtenus en échantillonnant les gains moyens des bras d'une manière uniforme dans l'en-

Strategies	B1	B2	B3	B4	B5	B6	B7	B8	B9
0.00-greedy	**1.7**	**2.5**	**3.0**	144.1	245.8	249.2	329	466.4	484.5
0.05-greedy	54.8	66.9	81.2	81.9	147.0	**232.7**	**166.8**	300.1	395.6
0.10-greedy	109.6	132.4	160.6	117.8	191.7	279.4	**186.8**	317.8	408.3
0.15-greedy	165	195.8	240.6	158.2	238.5	336.4	216.5	332.8	460.5
0.20-greedy	218.6	262.1	323.2	203.0	302.9	408.2	249.9	385.6	516.9
0.25-greedy	274.5	329.5	399.8	251.2	361.1	469.7	289.8	434.0	580.6
0.30-greedy	326.5	394.0	482.1	296.9	423.3	543.8	331.3	489.0	639.5
0.35-greedy	382.8	458.5	561.5	342.8	483.5	615.3	369.3	542.3	704.7
0.40-greedy	437.9	525.6	638.1	390.0	548.1	686.4	410.1	596.9	764.4
0.45-greedy	491.9	589.6	722.4	440.0	609.0	767.7	455.5	654.6	840.1
1-Dgreedy	3.2	4.1	4.7	**98.7**	248.5	**245.8**	235.9	443.2	476.1
20-Dgreedy	30.4	32.6	36.7	**59.2**	**101.7**	**210.2**	**149.9**	295.1	404.7
40-Dgreedy	56.1	65.4	70.3	71.5	120.5	**194.5**	**152.6**	**253.4**	**363.7**
60-Dgreedy	80.6	93.9	104.1	84.3	146.9	214	**158.5**	**254.3**	**344.0**
80-Dgreedy	103.6	120.7	135.9	102.0	151.6	231.1	**167.3**	**246.3**	371.7
120-Dgreedy	144.8	167.8	195.7	135.6	193.0	280.8	184.5	**269.0**	379.4
160-Dgreedy	180.9	212	249.5	163.2	235.1	325.2	203.4	294.6	434.1
200-Dgreedy	213	254.6	302.0	192.6	274.9	365.8	228.1	324.0	472.7
DP-*greedy*	**1.7**	**2.5**	**3.0**	**42.2**	**113.3**	**204.5**	**149.8**	**225.5**	**371.2**

TABLE 6.4 – Le regret cumulé au round $H = 4000$ pour les dix-neuf stratégies sur les neuf premiers problèmes (de B1 à B9).

semble $[0, 1]$. Les cas-tests diffèrent par le nombre de bras disponibles et par l'écart-type des gains des bras (voir table 6.3). Pour chaque tâche, les gains obtenus après avoir testé un bras suivent une distribution de type normale ($\mathbf{z}_k \sim N[\mu_k, \sigma_k]$). Les performances des algorithmes de *bandit* sont obtenues en faisant la moyenne des regrets cumulés sur l'ensemble des cent tâches.

Pour les trois cas-tests générés à partir de données réelles, nous utilisons les mêmes données que celles de la section précédente où un programme doit interroger les pages d'accueil de sites internet de différentes universités et où le but est de minimiser le temps d'attente. Dans nos expériences, pour générer un grand nombre de problèmes, nous avons sélectionné cent fois $K = 3$, $K = 5$, ou $K = 10$ universités et calculé les performances des stratégies sur un horizon de 4000 étapes. Ces trois problèmes sont dénotés B13, B14 et B15.

Les tableaux 6.4 et 6.5 ainsi que la figure 6.2 contiennent les résultats expérimentaux. Les tableaux donnent, pour chacune des dix-neuf stratégies et pour les quinze problèmes, la moyenne des regrets cumulés sur les cent répétitions lorsque l'horizon vaut $H = 4000$. Selon un t-test jumelé, une valeur en gras dans les tableaux signifie que pour ce problème, la moyenne du regret cumulé de la stratégie n'est pas significativement différente ($p > 0.01$) du regret de la meilleure stratégie.

Les deux résultats suivants sont observés :

- excepté pour le problème numéro $B12$, il n'y a pas de différence significative entre les performances de DP-*greedy* et celles de la meilleure stratégie,
- dans huit problèmes sur quinze, la stratégie DP-*greedy* est la meilleure stratégie parmi toutes les autres stratégies semi-uniformes.

Strategies	B10	B11	B12	B13	B14	B15
0.00-greedy	599.3	534.9	603.6	**269216.5**	**218262.2**	**703681.6**
0.05-greedy	281.9	384.9	559.9	304536.1	279927.3	632359.4
0.10-greedy	270.7	382.5	561	397862.4	415458.7	910185.1
0.15-greedy	300.1	401.8	592.4	483358.0	557590.3	1159969.6
0.20-greedy	323.4	446.8	628.1	598526.7	701810.4	1429159.4
0.25-greedy	349.0	490.2	684.3	711834.9	855150.9	1716417.5
0.30-greedy	383.5	541.1	743.8	842439.0	999679.3	1992845.8
0.35-greedy	424.1	587.3	796.8	972912.6	1157161.6	2254550.4
0.40-greedy	464.2	639.2	839.9	1095672.3	1308751.1	2572096.9
0.45-greedy	508.2	693.0	911.6	1225084.2	1464515.5	2853342.4
1-Dgreedy	569.5	528.4	597.4	**271872.5**	**212768.5**	**694380.8**
20-Dgreedy	240.3	394.4	**543.5**	**241194.0**	**209618.8**	**463682.9**
40-Dgreedy	**215.1**	**347.8**	**532.5**	294683.3	268010.1	566396.1
60-Dgreedy	**216.1**	**331.4**	**514.1**	**286044.7**	320430.4	687074.8
80-Dgreedy	**212.2**	**326.0**	**497**	305760.5	368216.9	793232.9
120-Dgreedy	**216.1**	**334.8**	**510.7**	380555.2	479727.5	978683.0
160-Dgreedy	237.7	**365.4**	**519.6**	468543.4	576326.0	1153664.9
200-Dgreedy	256.6	395.6	557.1	551275.7	670023.0	1327895.8
DP-*greedy*	**209.3**	**322.7**	585.9	**267489.2**	**224806.0**	**675138.0**

TABLE 6.5 – Le regret cumulé au round $H = 4000$ pour les dix-neuf stratégies sur les six derniers problèmes (de B10 à B15).

L'utilisation de l'algorithme DP-*greedy* se rapproche donc de l'utilisation d'une stratégie semi-uniforme ϵ-*greedy* classique avec un paramètre ϵ optimal. En d'autres termes, DP-*greedy* surpasse les techniques connues jusqu'ici puisqu'il atteint les performances des autres stratégies semi-uniformes avec un paramètre optimal ϵ choisi a posteriori. Le succès de DP-*greedy* s'explique par les capacités d'adaptation de la stratégie, qui ajuste automatiquement le niveau d'exploration/exploitation en fonction de la difficulté du problème et de l'état actuel.

Des informations supplémentaires sur le comportement de la stratégie DP-*greedy* sont données dans la figure 6.2. Cette figure présente les courbes d'évolution du pourcentage d'exploration réalisé durant les 4000 rounds ; les problèmes allant de B4 à B12. Il est à noter que pour les problèmes très simples (comme B1, B2 et B3), la stratégie se comporte comme une stratégie de pure exploitation. La figure 6.2 montre que sur des problèmes plus complexes (c'est-à-dire lorsque K ou σ est grand), DP-*greedy* augmente tout d'abord automatiquement le taux d'exploration et, lorsque plus d'observations sont collectées, la stratégie bascule vers davantage d'exploitation.

6.5 Conclusions

Ce chapitre a proposé deux nouvelles stratégies semi-uniformes pour résoudre le MABP : ϵ-PCSgreedy et DP-*greedy*. Dans les deux cas, nous utilisons des notions déjà introduites dans le chapitre 4.

La stratégie ϵ-PCSgreedy propose de remplacer l'exploration aléatoire uniforme par une exploration utilisant la quantité $\Pr\{S_{[K]}\}$ pour choisir

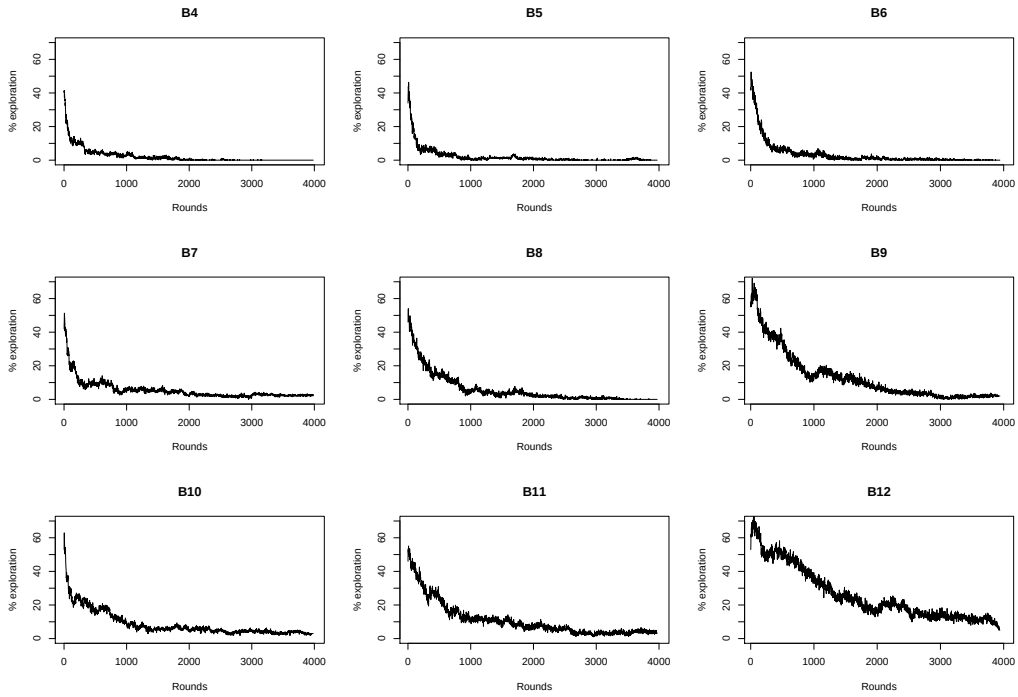


FIGURE 6.2 – Evolution du pourcentage d'exploration durant les 4000 rounds de DP-greedy sur les problèmes synthétiques. Le pourcentage d'exploration dans B1, B2 et B3 est toujours nul.

l'alternative à tester. Les résultats expérimentaux montrent qu'il y a une amélioration significative des performances de ϵ-PCSgreedy.

La deuxième stratégie proposée est la stratégie DP-*greedy*. Elle est construite sur le concept de μ_g et utilise des méthodes de programmation dynamique pour poser la meilleure action possible. A la différence des stratégies classiques, DP-*greedy* ne choisit pas directement l'alternative à tester mais plutôt le type d'action à effectuer : exploitation gloutonne ou exploration aléatoire. Nous avons montré que la quantité μ_g joue un rôle important dans la définition de cette stratégie. Puisque PSI n'est pas une supposition réaliste dans le MABP, nous avons dû utiliser l'un des estimateurs proposés dans la section 4.4. Le fait de disposer d'estimateurs de μ_g a permis de proposer une version de la stratégie pouvant être utilisée dans le cas ISI. Une série d'expériences sur des données réelles et synthétiques ont montré que DP-*greedy* est une stratégie efficace comparée aux autres stratégies semi-uniformes.

Chapitre 7

Application des algorithmes d'apprentissage supervisé en anesthésie – contributions

Ces dernières années, de plus en plus d'organisations dans différents domaines allouent d'importantes ressources pour développer et maintenir de vastes bases de données. Les institutions médicales ne font pas exception à la règle. Aujourd'hui, de nombreuses équipes dans le milieu médical utilisent couramment des systèmes informatiques pour lire et enregistrer différents signaux en provenance du patient. Dans ce contexte, les méthodes d'apprentissage artificiel et d'analyse de données constituent des outils permettant d'extraire de l'information utile dans le but d'améliorer les techniques de diagnostic, de détecter des événements importants ou encore de développer des systèmes d'aide à la décision.

Cette thèse a été financée par un projet FIRST Europe de la Région Wallonne en collaboration avec le service d'anesthésie de l'Hôpital Erasme [1] et la société Mexys SA [2]. Certains algorithmes proposés dans cette thèse ont été appliqués sur des données réelles provenant du milieu hospitalier.

Différents systèmes informatiques ont récemment été développés pour assister l'anesthésiste dans son travail quotidien en bloc opératoire. Par exemple, le système ITB (*Infusion Toolbox*) [23] – implémenté en C et en Smalltalk [3] – est un logiciel utilisé depuis de nombreuses années par le service d'anesthésie de l'Hôpital Erasme.

Ce système surveille l'état du patient et contrôle le débit des pompes

1. Erasme est l'hôpital universitaire lié à l'Université Libre de Bruxelles.
2. La société Mexys SA – basée à Mons (Belgique) – est une société spécialisée dans la conception et la réalisation de produits liés à l'informatique médicale. Plus particulièrement, Mexys a développé un système de rapports d'anesthésie informatisés dans le domaine de l'anesthésie. Plus d'informations peuvent être obtenues sur leur site internet `http://www.mexys.com/`
3. http ://www.smalltalk.org/

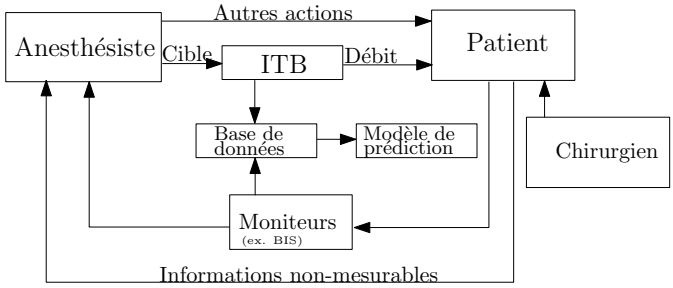


FIGURE 7.1 – Le logiciel ITB accomplit deux tâches principales. (i) En fonction du niveau de concentration cible des drogues choisi par l'anesthésiste, ITB fixe leur débit. (ii) ITB enregistre également, dans une base de données, les valeurs des signaux des moniteurs et les actions des anesthésistes.

contenant les différentes drogues (voir la figure 7.1) et ceci via des modèles pharmacocinétiques et pharmacodynamiques [8]. Durant l'intervention, le système ITB enregistre :(i) différentes statistiques du patient (âge, poids,...), (ii) certaines actions de l'anesthésiste (modification de concentration des drogues) et (iii) les valeurs des moniteurs (rythme cardiaque, BIS, ...).

Dans ce chapitre, nous étudions l'impact de la concentration du propofol dans le cerveau, qui agit sur la profondeur d'inconscience du patient. Ce niveau d'inconscience est mesuré par un moniteur d'index bispectral (BIS) [78, 39]. Le propofol est une drogue hypnotique à action courte injectée par voie intraveineuse. Cette drogue est utilisée pour induire et maintenir une anesthésie générale.

Nous allons analyser et évaluer le rôle des techniques d'apprentissage artificiel afin d'extraire de l'information utile, à partir de la base de données générée par le système ITB. Sur base d'informations sur l'état du patient et connaissant le niveau de concentration de propofol, nous allons proposer un système d'aide à la décision qui fournit une estimation quant à l'évolution future du niveau du BIS, ce qui aidera l'anesthésiste à prendre des décisions concernant le niveau de drogue optimal. L'algorithme exploreMaxμ_g – présenté dans la section 5.3 – sera également utilisé afin de réaliser une sélection de variables en réduisant au maximum le nombre d'erreurs de *leave-one-out*.

Ce chapitre contient les contributions de cette thèse en rapport avec l'application des techniques d'apprentissage artificiel en anesthésie. Il est structuré comme suit : la section 7.1 introduit les concepts d'anesthésie nécessaires à la compréhension de ce chapitre. Le problème de prédiction est formellement défini dans la section 7.2. La section 7.3 présente le logiciel d'aide à la décision. La sélection de variables est une phase importante dans la résolution du problème d'apprentissage. Dans la section 7.4, nous proposons d'utiliser l'algorithme exploreMaxμ_g afin d'accélérer la procédure de sélection de variables. Enfin, la dernière section 7.5 décrit les conclusions.

7.1 L'anesthésie intraveineuse à objectif de concentration

L'anesthésie générale [63] est un acte médical dont l'objectif principal est la suspension temporaire et réversible de la conscience et de la sensibilité douloureuse. L'anesthésie peut être pratiquée par agents *inhalés*, par voie *intraveineuse* ou encore par une combinaison des deux techniques ; nous parlons alors d'anesthésie *balancée*. Dans cette thèse, nous nous pencherons uniquement sur les anesthésies pratiquées par voie intraveineuse [79].

Différents agents anesthésiques peuvent être utilisés durant une intervention chirurgicale ; ils peuvent être classés en trois catégories : les analgésiques, les hypnotiques et les curares. Les agents d'analgésie ont pour but de diminuer le retentissement des actes douloureux. C'est le rémifentanil qui est utilisé dans cette étude comme agent analgésique. L'effet recherché par les agents hypnotiques est la perte de conscience et le maintien de cette inconscience. Dans ce travail, c'est le propofol qui est utilisé comme hypnotique. Enfin les curares, comme le cisatracurium utilisé ici, sont des agents qui provoquent une paralysie pour empêcher les mouvements nuisibles à la chirurgie ou faciliter ceux-ci en relâchant les muscles [4].

Cette classification des agents en trois catégories distinctes n'empêche pas une interaction entre eux ; par exemple une forte dose d'analgésique permet de diminuer le niveau de drogue hypnotique [77]. Il faut donc tenir compte de cette interaction. L'anesthésiste va devoir ajuster en permanence le niveau des différents agents afin d'amener puis de maintenir le patient dans l'état désiré durant l'entièreté de l'opération.

Afin d'aider les anesthésistes à maintenir les patients dans un état souhaité, la technique d'*Anesthésie IntraVeineuse à Objectif de Concentration* (AIVOC) [76] a vu le jour. L'objectif à atteindre avec l'AIVOC est de proposer un système où l'anesthésiste introduit la concentration désirée d'une drogue dans une zone effet (par exemple le cerveau) et c'est alors le système qui calcule les débits à injecter pour obtenir et maintenir la concentration désirée.

Les techniques sous-tendant cette procédure sont de type pharmacocinétiques multi-compartimentales [8]. Ces techniques cherchent à déterminer le débit de la perfusion à injecter au patient pour atteindre et maintenir la concentration désirée dans la zone effet. La figure 7.2 modélise un tel problème de pharmacocinétique multi-compartimental. Dans ce type de modèle, la drogue est injectée dans le compartiment central et se diffuse ensuite dans les autres compartiments. Ce modèle tricompartimental est paramétré par les volumes de ses compartiments (V_1, V_2, V_3), par ses constantes de transfert (k_{12}, k_{21}, k_{13}, k_{31},k_{1e}), ainsi que par ses constantes d'élimination k_{10} et k_{e0}. La concentration dans le compartiment central correspond à la

4. Le cisatracurium n'entre pas en compte dans la suite de ce travail.

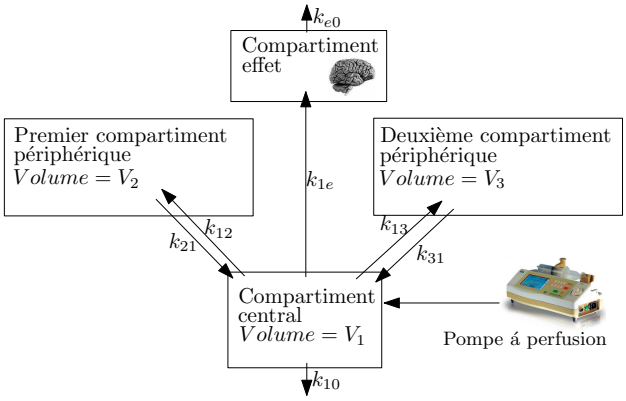


FIGURE 7.2 – Représentation d'un modèle pharmacocinétique tricompartimental associé à un compartiment effet.

concentration plasmatique. Les deux autres compartiments périphériques sont couramment désignés pour représenter les graisses et les muscles mais il ne s'agit que d'une interprétation des équations. Le dernier compartiment correspond à la zone effet (par exemple le cerveau). Dans ce modèle à compartiments, la drogue ne s'évacue que via le compartiment central ou le compartiment effet. Les valeurs des différents paramètres du modèle pharmacocinétique sont fixées à partir de mesures recueillies sur des patients volontaires. Il existe différents modèles pharmacocinétiques pour les différentes drogues. Dans cette étude, ITB ajuste le débit de propofol selon le modèle de Schnider [74] et le débit du rémifentanil selon le modèle de Minto [61].

Un bolus désigne une dose massive d'agent anesthésique injectée en une courte période de temps. Ceci aura pour effet de remplir le compartiment central. Ensuite, la drogue se propagera dans les deux compartiments périphériques et dans le compartiment effet. Il y a un transfert des compartiments périphériques vers le compartiment central lorsque la concentration dans un compartiment périphérique devient plus élevée que dans le compartiment central.

Voici comment Frédérique Servin décrit, dans l'introduction de l'un de ses articles [76], l'anesthésie intraveineuse à objectif de concentration :

≪ En anesthésie, l'effet pharmacologique d'un agent est fonction de la concentration de cet agent en face des récepteurs impliqués. Par ailleurs, l'effet est constant si la concentration est constante, et ce quelle que soit la durée pendant laquelle cette concentration est maintenue[...]. Ainsi, chercher à obtenir un effet revient à obtenir une concentration, et la connaissance de la zone de concentrations efficaces d'un agent chez un patient donné, comme on peut l'acquérir à l'induction de l'anesthésie, permet de conduire de façon plus précise l'ensemble de l'administration de l'agent. Dans cette optique, l'administration en débit massique n'est qu'un pis-aller.

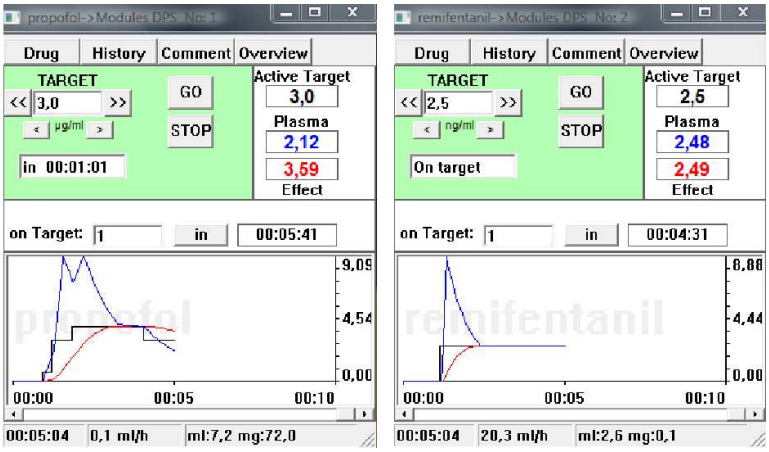


FIGURE 7.3 – Capture d'écran des deux principales fenêtres du logiciel ITB permettant de choisir les niveaux de concentration cibles du propofol et du rémifentanil.

Avec les techniques habituelles, obtenir instantanément une concentration souhaitée, donc un effet recherché est impossible du fait de la cinétique complexe, multicompartimentale, des agents intraveineux. Par définition, les injections intraveineuses directes en bolus provoquent un pic instantané de concentration plasmatique. La concentration diminue ensuite constamment jusqu'au bolus suivant. Pour pallier l'effet oscillant, lié aux bolus itératifs, l'entretien de l'anesthésie fait souvent appel à une perfusion continue. Malheureusement, si le débit de perfusion est constant, comme le transfert vers les compartiments périphériques diminue au cours du temps, puisqu'il dépend de la différence de concentration entre le sang et la périphérie, la concentration résultante augmente progressivement. Pour obtenir une concentration recherchée, il faut d'abord administrer un bolus calibré, puis prendre instantanément le relais par une perfusion à débit progressivement décroissant. Le seul moyen d'y parvenir, qui présente par ailleurs au quotidien l'avantage de rendre invisible l'étape de calcul pharmacocinétique, est de calculer précisément la dose pour obtenir la concentration à l'aide d'un microprocesseur.

L'anesthésie intraveineuse à objectif de concentration (AIVOC) représente ce mode d'administration où l'anesthésiste recherche une concentration (un effet), à charge à un programme de calcul validé de pourvoir au débit nécessaire pour l'obtenir. ≫

L'anesthésiste définit donc une concentration désirée des drogues au système informatique et c'est le système informatique qui prend en charge le calcul des débits pour atteindre la concentration désirée. C'est le logiciel appelé ITB [23] qui est utilisé à L'Hôpital Erasme comme système AIVOC. La figure 7.3 contient une capture d'écran des deux principales fenêtres du logiciel. La première fenêtre est utilisée pour choisir le niveau de concentration cible du propofol et la deuxième fenêtre est dédiée au rémifentanil.

L'effet d'un agent hypnotique (comme le propofol) est la perte de conscience. Différents moniteurs existent pour mesurer le niveau de la profondeur d'inconscience du patient, comme par exemple le moniteur d'entropie [47], l'index bispectral (BIS) [78, 39] ou encore le NeuroSENSE [90]. Tous ces moniteurs sont basés sur une analyse de l'électroencéphalogramme

FIGURE 7.4 – Le moniteur d'index bispectral (BIS) transforme l'électroencéphalogramme du patient en un scalaire compris entre les valeurs 0 et 100 (source : http ://www.aspectmedical.com/).

du patient. Dans le cadre de ce travail, nous nous limiterons à l'étude des facteurs qui influencent le signal BIS [5] (voir figure 7.4).

Le moniteur BIS mesure le niveau de la profondeur d'inconscience du patient via un scalaire compris entre les valeurs 0 et 100 où 0 correspond à un électroencéphalogramme plat et 100 correspond au signal d'un patient éveillé. Pour que le patient n'ait pas de souvenir de l'intervention, l'anesthésiste essayera généralement de maintenir le signal du BIS dans une fourchette de valeurs comprises entre 40 et 60.

La figure 7.5 est un exemple d'évolution classique du signal BIS. Le signal est – lorsque le patient est encore conscient – proche de 100 et chute rapidement aux alentours de 50 une fois l'induction commencée. Le signal est ensuite maintenu dans une fourchette de valeurs comprises entre 40 et 60 pour remonter lorsque l'anesthésiste arrête la diffusion de l'hypnotique. Ce moniteur permet à l'anesthésiste de détecter des situations de profondeur d'inconscience trop ou pas assez forte et par voie de conséquence d'adapter la titration des agents d'anesthésie afin d'éviter des situations dangereuses pour le patient.

Il est parfois difficile pour un anesthésiste – surtout lorsqu'il est inexpérimenté – de prédire l'évolution du signal BIS après une modification du niveau de propofol. Deux patients traités avec une dose similaire d'une même drogue et à une même phase de chirurgie peuvent manifester des réactions très variées. Ce problème est généralement dénoté dans la littérature scien-

5. Le moniteur d'index bispectral est commercialisé par la société Aspect Medical Systems `http://www.aspectmedical.com/`

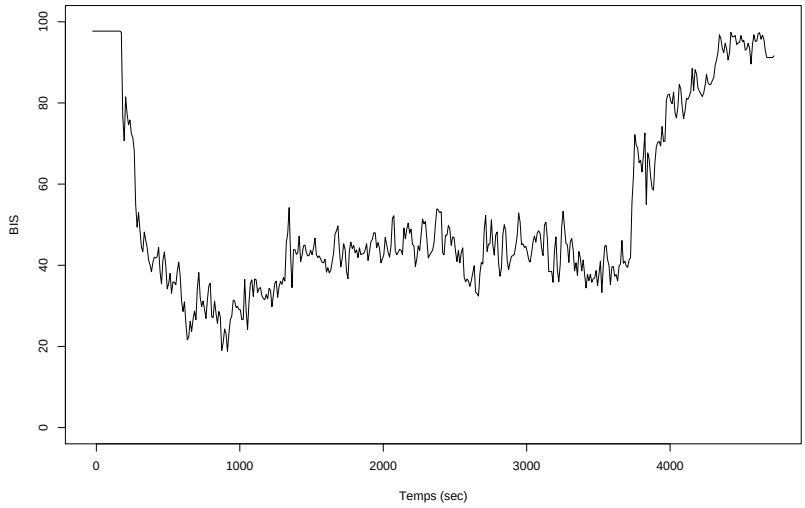


FIGURE 7.5 – Exemple d'évolution du signal BIS qui estime la profondeur d'inconscience du patient.

tifique par le *problème de variabilité interindividuelle*. Afin de pallier ce problème, nous avons proposé une technique où les dix premières minutes – après la première modification de propofol – sont consacrées à la construction de variables représentant la sensibilité du patient aux différentes drogues. Ces variables sont ensuite utilisées par notre modèle prédictif afin de s'adapter aux particularités du patient.

La figure 7.6 montre l'un des chariots utilisés à l'hôpital Erasme pour pratiquer des anesthésies de type AIVOC. A la base du chariot, trois pompes sont installées pouvant contenir l'hypnotique, l'analgésique et le curare. Au-dessus des pompes, se trouve le moniteur BIS. Et enfin, sur la partie supérieure du chariot, siège l'ordinateur sur lequel ITB est configuré.

7.2 Formalisation du problème

La figure 7.7 est composée de deux graphiques concernant l'évolution du BIS sur un court laps de temps après une modification de la concentration de propofol. Dans l'exemple de gauche, le patient est un homme âgé de 70 ans, pesant 76 kg et mesurant 174 cm. 412 secondes après le début de l'intervention, le niveau de concentration de propofol change pour passer de $0.5\mu g/ml$ à $2\mu g/ml$ pendant que le rémifentanil (analgésique) est à $5ng/ml$. Après cette modification du niveau de concentration de propofol, la profondeur d'inconscience du patient augmente ce qui se traduit par une diminution du BIS. L'exemple de droite concerne un homme âgé de 56 ans,

FIGURE 7.6 – Exemple d'un chariot utilisé à l'hôpital Erasme, composé de trois pompes pour les différentes drogues, d'un moniteur d'inconscience BIS et d'un ordinateur sur lequel ITB est configuré.

pesant 64 kg et mesurant 170 cm. C'est un exemple de fin de chirurgie où, après 14490 secondes (± 4 heures), l'anesthésiste place le niveau de propofol à zéro (l'ancienne valeur étant de $2\mu g/ml$) ce qui a pour conséquence une remontée rapide du BIS.

Cette évolution du signal BIS est parfois difficile à prédire. Dans le cadre de cette thèse, nous avons développé un prototype de système d'aide à la décision afin d'assister l'anesthésiste lors de l'ajustement du niveau de concentration de propofol pour que le niveau du BIS du patient atteigne une valeur désirée. Pour ce faire, nous cherchons à produire une hypothèse prédictive capable d'estimer l'évolution du BIS dans une fenêtre temporelle de dix minutes après la modification d'une concentration cible de propofol. Cette section décrit la procédure d'apprentissage utilisée afin d'estimer l'hypothèse prédictive du BIS à partir des échantillons produits par le logiciel ITB.

La valeur du BIS sera prédite à des intervalles réguliers de Δ secondes à partir de l'instant t où le propofol a été modifié. De cette manière, la problématique est décomposée en un ensemble de sous-problèmes de prédiction : un sous-problème de prédiction pour chaque instant $t + T$ où $T \in \{\Delta, 2\Delta, \ldots, S\Delta\}$. Une fois ces S hypothèses prédictives construites, les prédictions sont assemblées pour former une courbe de l'évolution es-

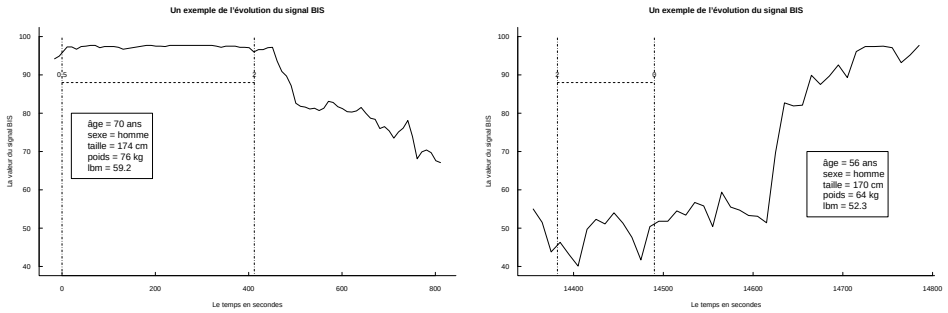


FIGURE 7.7 – Deux exemples de l'évolution du BIS sur une courte période. Dans le graphique de gauche, le niveau de concentration de propofol passe de 0.5 à 2 et l'on observe une chute du signal BIS. Dans l'autre graphique, le niveau de concentration de propofol est placé à zéro ce qui a pour conséquence que ITB arrête d'injecter du propofol et l'on observe une remontée du signal BIS.

timée du BIS dans la fenêtre temporelle $[t + \Delta, t + S\Delta]$.

Supposons que la dynamique de l'index BIS puisse être décrite par

$$B\left(t+T\right) = f^{T}\left(B\left(t\right), X\left(t\right)\right) + \epsilon^{T}(t), \qquad T = \Delta, 2\Delta, \ldots, S\Delta \quad (7.1)$$

où t dénote un instant de l'intervention où la concentration cible de propofol a été modifiée, $t = 0$ correspond à l'instant de la première modification de propofol, $\epsilon^{T}(t)$ est un terme d'erreur, $B\left(t\right)$ est la valeur du BIS à l'instant t et $X(t)$ est un vecteur dont le contenu est décrit dans le tableau 7.1. Il est à noter que dans notre configuration, nous avons fixé $S = 20$ et $\Delta = 30$ secondes. Il y a donc vingt hypothèses permettant de décrire l'évolution du BIS dans une fenêtre temporelle de 10 minutes.

7.2.1 L'ensemble d'apprentissage

Le tableau 7.1 contient, pour chaque variable de $X\left(t\right)$, son symbole, une courte description et les deux dernières colonnes spécifient si la variable est constante au cours de l'opération et si la variable est créée durant l'initialisation.

Nous entendons par initialisation, la phase de dix minutes qui commence après la première modification de la cible de concentration de propofol [6]. Les variables créées durant cette phase mesurent essentiellement la réaction du patient aux différentes drogues durant les premiers instants de l'intervention.

6. C'est pendant ces dix premières minutes de l'induction (phase de préparation du patient à la chirurgie) qu'a lieu la perte de conscience. L'induction est une bonne occasion pour mesurer, via différentes variables, la réactivité du patient aux différentes drogues.

Variable	Définition	Constante	Initialisation
$\Delta p(t)$	Ampleur de la modification de la concentration de propofol ($\mu g/ml$) à l'instant t.		
$p(t)$	Niveau de concentration de propofol ($\mu g/ml$) avant la modification.		
$\Delta timeP$	Temps (sec) entre la dernière modification de propofol et t.		
t	Instant (sec) où a lieu la modification de propofol.		
$r(t)$	Niveau de concentration de rémifentanil (ng/ml) à l'instant t.		
$T_r(t)$	Temps (sec) entre la dernière modification de rémifentanil et t.		
a	Age du patient.	✓	
s	Sexe du patient.	✓	
he	Taille du patient (cm).	✓	
w	Poids du patient (kg).	✓	
lbm	*Lean Body Mass* (LBM) est égal à $\begin{cases} 1.1 \cdot poids - 128\frac{poids^2}{taille^2} & \text{if } sexe = \text{homme} \\ 1.07 \cdot poids - 148\frac{poids^2}{taille^2} & \text{if } sexe = \text{femme} \end{cases}$	✓	
$p80$	Niveau de propofol ($\mu g/ml$) lorsque le BIS a atteint 80.	✓	✓
$p70$	Niveau de propofol ($\mu g/ml$) lorsque le BIS a atteint 70.	✓	✓
$p60$	Niveau de propofol ($\mu g/ml$) lorsque le BIS a atteint 60.	✓	✓
μ_r	Niveau de concentration moyen de rémifentanil (ng/ml).	✓	✓
μ_p	Niveau de concentration moyen de propofol ($\mu g/ml$).	✓	✓
max_p	Maximum atteint par la concentration de propofol ($\mu g/ml$).	✓	✓
$tMax_p$	Instant (sec) où le propofol est à son maximum.	✓	✓
min_B	Niveau maximum du BIS.	✓	✓
$tMin_B$	Instant (sec) où le BIS est à son maximum.	✓	✓
R	Rapport entre max_p et min_B.	✓	✓

TABLE 7.1 – Ce tableau décrit les variables que contient le vecteur $X(t)$ de l'équation (7.1).

Ces variables doivent aider l'hypothèse prédictive à s'adapter à la variabilité interindividuelle.

Nous allons maintenant approfondir certaines variables du tableau 7.1. La variable $p(t)$ est le niveau de la concentration de propofol (en $\mu g/ml$) avant la modification à l'instant t, et $\Delta p(t)$ est la modification de la concentration de propofol (en $\mu g/ml$). En d'autres termes, la nouvelle concentration de propofol à l'instant t est égale à $p(t) + \Delta p(t)$. La variable $r(t)$ mesure le niveau de concentration du rémifentanil (en ng/ml) lorsque le propofol est modifié (c'est-à-dire à l'instant t) et $T_r(t)$ est le temps (en secondes) depuis lequel le rémifentanil n'a pas été modifié. Les quinze variables suivantes sont constantes, c'est-à-dire qu'une fois leurs valeurs fixées, elles ne sont plus modifiées au cours du temps. Leur but est de traiter la variabilité in-

	$D_N^{\Delta t=30}$	$D_N^{\Delta t=60}$	$D_N^{\Delta t=90}$	$D_N^{\Delta t=120}$	$D_N^{\Delta t=150}$
$N =$	1786	1656	1542	1465	1390
	$D_N^{\Delta t=180}$	$D_N^{\Delta t=210}$	$D_N^{\Delta t=240}$	$D_N^{\Delta t=270}$	$D_N^{\Delta t=300}$
$N =$	1333	1270	1244	1195	1148
	$D_N^{\Delta t=330}$	$D_N^{\Delta t=360}$	$D_N^{\Delta t=390}$	$D_N^{\Delta t=420}$	$D_N^{\Delta t=450}$
$N =$	1101	1060	1017	986	957
	$D_N^{\Delta t=480}$	$D_N^{\Delta t=510}$	$D_N^{\Delta t=540}$	$D_N^{\Delta t=570}$	$D_N^{\Delta t=600}$
$N =$	917	890	860	824	801

TABLE 7.2 – Pour les 20 ensembles d'apprentissage, ce tableau indique le nombre d'échantillons disponibles.

terindividuelle en ajustant l'hypothèse à chaque patient. Les cinq premières constantes décrivent le patient de manière conventionnelle. Les dix suivantes sont créées lors de la phase d'initialisation. Les quantités $pb80$, $pb70$ et $pb60$ sont les niveaux de concentration de propofol lorsque le BIS est respectivement à 80, 70 et 60. μ_r et μ_p mesurent le niveau moyen de concentration du rémifentanil et du propofol durant les dix premières minutes de l'intervention. max_p est le niveau de concentration maximum du propofol utilisé par l'anesthésiste durant l'initialisation et $tMax_p$ est l'instant où max_p a été atteint. min_B est le niveau du BIS minimum et $tMin_B$ est l'instant où min_B a été atteint. Finalement, R est le rapport entre la variable max_p et la variable min_B. Une valeur élevée de R signifie que l'induction a été agressive c'est-à-dire que la concentration de propofol cible était élevée et que le BIS est descendu à une très petite valeur.

La fonction $f^T()$ donne (à un facteur d'erreur $\epsilon^T(t)$ près) le niveau du BIS futur à l'instant $t + T$ (avec $T \geq 0$) où t correspond à un instant de modification du niveau de concentration de propofol. Cette fonction est en réalité inaccessible et doit être estimée.

A partir de la base de données de ITB, les informations de 1069 interventions chirurgicales ont été utilisées pour construire vingt ensembles d'apprentissage $\left\{D_N^{T=30}, \ldots, D_N^{T=600}\right\}$. Dans l'ensemble $D_N^{T=\vartheta}$, un échantillon d'apprentissage est créé si à un instant t (avec $t > 600$sec ; c'est-à-dire après la phase d'initialisation) le niveau de concentration cible de propofol est modifié par l'anesthésiste et qu'il n'y a pas d'autre modification de propofol dans l'intervalle $[t, t + \vartheta]$. Le tableau 7.2 indique, pour chaque ensemble d'apprentissage, le nombre d'échantillons disponibles.

7.2.2 La sélection de variables

Les procédures de sélection de variables (voir section 2.5 à la page 34) ne sont pas uniquement utiles d'un point de vue statistique pour améliorer les performances de l'hypothèse sélectionnée, mais elles constituent également

T	Variable sélectionnée par l'algorithme SFS où Λ correspond à la famille des hypothèses linéaires.
30sec	$B(t) + \Delta p(t) + p(t) + min_B + r(t) + p70 + \Delta timeP + R + w + tMin_B + p60 + he + lbm$
60sec	$B(t) + \Delta p(t) + p(t) + min_B + p70 + p80 + \Delta timeP + t + r(t)$
90sec	$B(t) + \Delta p(t) + p(t) + min_B + t + \Delta timeP + w + p70 + R$
120sec	$B(t) + \Delta p(t) + p(t) + min_B + p70 + \Delta timeP + t + R + max_p rop$
150sec	$B(t) + \Delta p(t) + p(t) + min_B + tMin_B + p70 + R + t + \Delta timeP + he$
180sec	$B(t) + \Delta p(t) + p(t) + min_B + \mu_r + \Delta timeP + tMin_B + p70 + R + p80 + lbm + age$
210sec	$B(t) + \Delta p(t) + p(t) + min_B + \mu_p + R + t + \Delta timeP + r(t) + he + max_p rop + p60$
240sec	$B(t) + \Delta p(t) + p(t) + min_B + p70 + R + t + \Delta timeP$
270sec	$B(t) + \Delta p(t) + p(t) + min_B + p70 + R + he + t + \Delta timeP + r(t)$
300sec	$B(t) + \Delta p(t) + p(t) + min_B + p70 + p80 + \Delta timeP + tMax_p rop + t + R + max_p rop$
330sec	$B(t) + \Delta p(t) + p(t) + min_B + p70 + p80 + \mu_r + \Delta timeP + t + r(t) + R + max_p rop + p60$
360sec	$B(t) + \Delta p(t) + p(t) + min_B + p70 + R + max_p rop + t + r(t) + \Delta timeP + \mu_r$
390sec	$B(t) + \Delta p(t) + p(t) + min_B + r(t) + p70 + R + p80$
420sec	$B(t) + \Delta p(t) + p(t) + min_B + p70 + p80 + \mu_r$
450sec	$B(t) + \Delta p(t) + p(t) + min_B + max_p rop + p80 + R + \mu_r + p70$
480sec	$B(t) + \Delta p(t) + p(t) + min_B + p60 + R + t + \Delta timeP + \mu_r + p80 + max_p rop$
510sec	$B(t) + \Delta p(t) + p(t) + t + min_B + p70 + R + max_p rop + \Delta timeP + \mu_r + p80$
540sec	$B(t) + \Delta p(t) + p(t) + t + p70 + R + p60 + p80 + \mu_r$
570sec	$B(t) + \Delta p(t) + p(t) + p70 + p80 + min_B + \mu_p + t + R + \Delta timeP + w + p60 + \mu_r + age$
600sec	$B(t) + \Delta p(t) + p(t) + p70 + p80 + min_B + t + \Delta timeP + \mu_p + sexe + age + R + p60 + \mu_r$

TABLE 7.3 – Pour chaque ensemble d'apprentissage, ce tableau donne les variables d'entrée sélectionnées par l'algorithme SFS.

un bon moyen pour renvoyer de l'information à l'anesthésiste sur les variables qui jouent un rôle important dans l'évolution du BIS du patient.

Vu le grand nombre de variables (voir tableau 7.1) et afin d'améliorer les prédictions, nous avons – dans un premier temps – appliqué l'algorithme SFS (voir algorithme 1 à la page 35) sur les vingt ensembles d'apprentissage disponibles où l'ensemble Λ correspond à la famille des hypothèses linéaires. Utiliser la famille des hypothèses linéaires a l'avantage de permettre de calculer très rapidement l'erreur de *leave-one-out* via la PRESS (vois section B.2 à la page 171).

Le tableau 7.3 expose, pour chaque problème de prédiction, les variables sélectionnées par l'algorithme SFS lorsque la famille des hypothèses linéaires est utilisée pour calculer les erreurs de *leave-one-out*. Parmi les variables sélectionnées, certaines sont plus souvent choisies. Elles sont donc supposées apporter plus d'informations sur l'évolution du BIS. Par exemple, les variables $B(t)$, $\Delta p(t)$ et $p(t)$ sont chaque fois sélectionnées. A l'exception du cas $T = 540$sec, la variable min_B est également toujours sélectionnée et en-

T	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_0
30sec	0.904	−1.467	−0.83	0.013	0.447	−5.772	−1.14e−4	−4.10e−7	5.078
60sec	0.845	−2.717	−1.12	0.020	0.512	−5.177	−2.23e−4	9.61e−5	7.087
90sec	0.808	−4.284	−1.23	0.021	0.543	−9.927	−2.93e−4	1.55e−4	8.714
120sec	0.767	−5.281	−1.74	0.024	0.913	−11.231	−3.16e−4	9.87e−5	11.103
150sec	0.749	−6.086	−1.70	0.024	0.816	−13.997	−2.31e−4	1.26e−4	11.849
180sec	0.738	−6.208	−1.60	0.022	0.762	−15.215	−1.95e−4	5.79e−5	13.064
210sec	0.736	−6.501	−1.80	0.022	0.871	−12.952	−2.92e−4	1.52e−4	12.508
240sec	0.729	−7.380	−2.07	0.028	1.048	−14.063	−2.61e−4	1.24e−4	12.969
270sec	0.721	−7.745	−1.86	0.018	1.328	−19.085	−1.97e−4	9.88e−4	12.889
300sec	0.717	−8.343	−2.32	0.038	1.337	−16.643	−3.18e−4	9.34e−4	13.584
330sec	0.705	−9.018	−2.21	0.033	1.114	−11.787	−2.56e−4	1.10e−4	13.914
360sec	0.701	−9.371	−2.11	0.023	1.132	−13.220	−1.94e−4	1.07e−4	14.347
390sec	0.688	−9.652	−2.27	0.031	0.994	−16.703	−6.10e−5	1.75e−5	15.589
420sec	0.698	−9.690	−2.33	0.042	0.893	−9.448	−2.12e−4	5.25e−5	14.843
450sec	0.667	−9.302	−2.28	0.042	0.953	−8.704	−1.85e−4	8.56e−5	15.640
480sec	0.667	−9.703	−2.34	0.041	1.056	−13.438	−3.02e−4	1.85e−4	15.535
510sec	0.648	−10.508	−2.76	0.021	1.534	−23.173	−3.38e−4	2.36e−4	17.385
540sec	0.645	−10.185	−2.69	0.018	1.415	−18.751	−1.72e−4	1.82e−4	17.933
570sec	0.645	−10.047	−3.17	0.028	1.830	−16.700	−1.39e−4	1.67e−4	17.428
600sec	0.642	−10.832	−3.51	0.031	1.877	−16.833	−1.71e−4	1.97e−4	17.739

TABLE 7.4 – Les paramètres des vingt hypothèses linéaires de type (7.2).

fin les variables $p70$, R, $\Delta timeP$ et t sont fréquemment sélectionnées. Ces huit variables sont donc considérées comme le sous-ensemble de variables d'entrée contenant le plus d'informations.

La procédure de sélection de variables confirme l'importance de certaines variables pour prédire le BIS futur comme par exemple $B(t)$, $\Delta p(t)$ et $p(t)$ qui sont chaque fois sélectionnés. La majorité des variables constantes (âge, taille, ...) n'ont pas été sélectionnées. Ceci peut s'expliquer par leur intégration dans le modèle de pharmacocinétique du propofol utilisé par ITB. Il est à noter que trois des huit variables sont calculées durant les dix premières minutes de l'intervention. Ceci est un argument en faveur de notre contribution qui consiste à utiliser les premiers instants de l'intervention pour construire des variables supplémentaires mesurant la sensibilité du patient aux différentes drogues.

7.3 Le logiciel BisPrediction

Nous avons développé – dans le cadre de cette thèse – un prototype de système d'aide à la décision dénommé « *BisPrediction* » . Il estime l'évolution du signal BIS afin d'assister l'anesthésiste lors de l'ajustement du niveau de concentration de propofol.

Vingt hypothèses seront créées pour estimer l'évolution du signal BIS sur une période de dix minutes. Dans le cas de BisPrediction, les vingt hypothèses appartiennent à la famille des hypothèses linéaires et sont du

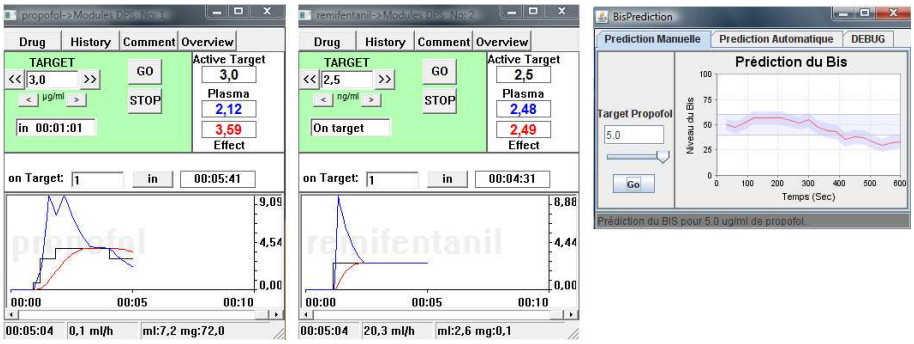


FIGURE 7.8 – Captures d'écran de ITB et BisPrediction (mode manuel)

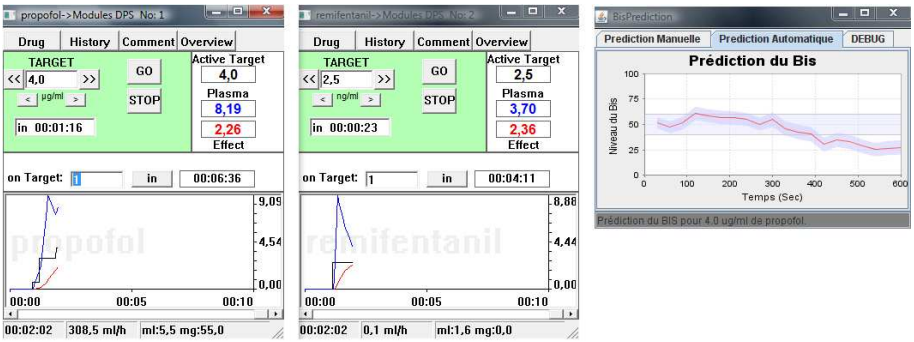


FIGURE 7.9 – Captures d'écran de ITB et BisPrediction (mode automatique)

type suivant :

$$\begin{aligned} h(\cdot, \alpha(T)) =& a_1 \cdot B(t) + a_2 \cdot \Delta p(t) + a_3 \cdot p(t) + \\ & a_4 \cdot min_B + a_5 \cdot p70 + a_6 \cdot R + \\ & a_7 \cdot \Delta timeP + a_8 \cdot t + a_0 \end{aligned} \tag{7.2}$$

où les valeurs des coefficients a_i, dépendantes de T, sont données dans le tableau 7.4. Chaque fois que l'anesthésiste modifie la concentration cible de propofol, le logiciel BisPrediction interagit avec ITB afin d'obtenir les valeurs des huit variables d'entrée.

Le logiciel à deux modes de fonctionnement. Les figures 7.8 et 7.9 contiennent des captures d'écran du logiciel BisPrediction en mode manuel et automatique. Les deux premières fenêtres sont les fenêtres classiques du logiciel ITB, et la troisième fenêtre correspond à BisPrediction.

Le logiciel BisPrediction reçoit toutes les informations sur l'état du patient via un serveur fonctionnant en arrière-plan. Cette communication est totalement transparente pour l'anesthésiste et permet de développer une interface graphique très épurée qui minimise les interactions entre BisPrediction et l'utilisateur.

Les deux onglets de BisPrediction permettent de choisir le mode de fonctionnement : mode manuel (figure 7.8) ou mode automatique (figure 7.9). Le troisième onglet – utilisé à des fins de débogage – affiche l'état des variables internes des fonctions prédictives.

En mode manuel, l'utilisateur choisit la cible de propofol, clique sur le bouton « Go » et un graphique présentant l'évolution estimée du BIS est affiché. Ceci permet d'estimer l'effet d'une modification de concentration cible de propofol avant de l'appliquer.

En mode automatique, aucune interaction n'est nécessaire avec l'interface graphique. Chaque fois que le niveau de concentration de propofol est modifié dans ITB (la première fenêtre partant de la gauche dans la figure 7.9), un signal est envoyé à BisPrediction – via le serveur fonctionnant en arrière-plan – pour l'affichage d'un graphique de l'estimation du BIS futur.

BisPrediction est implémenté avec le langage Java/Swing [7]. La librairie JFreeChart [8] est utilisée pour afficher la courbe du BIS et la technologie Java Native Interface (JNI) est utilisée pour interagir avec le serveur de communication tournant en tâche de fond.

Une étude à petite échelle a déjà été réalisée sur un groupe de cinq patients et a montré la faisabilité d'une telle approche. Un protocole d'analyse à plus grande échelle est en cours de rédaction et visera à mettre en place une étude randomisée en double aveugle afin de mesurer le gain éventuel que peut apporter un tel système d'aide à la décision sur la pratique quotidienne de l'anesthésiste.

7.4 Accélération de la procédure de sélection de variables avec exploreMaxμ_g

Si d est le nombre de variables d'entrée et si N est le nombre d'échantillons dans l'ensemble d'apprentissage D_N alors l'algorithme SFS (voir algorithme 1 à la page 35) utilise $N \times d$ estimations d'erreur de *leave-one-out* pour sélectionner la première meilleure variable. $N \times (d-1)$ erreurs de *leave-one-out* sont ensuite utilisées pour choisir la deuxième meilleure variable et ainsi de suite. Au total, l'algorithme SFS utilise $N \times d + N \times (d-1) + \ldots + N \times 2 = N \cdot \left(\frac{d(d+1)}{2} - 1\right)$ estimations d'erreur de *leave-one-out* pour sélectionner le sous-ensemble de variables optimales. Dans notre étude, il faut en plus calculer le sous-ensemble de variables optimales de S hypothèses différentes, soit un total de $S \cdot N \cdot \left(\frac{d(d+1)}{2} - 1\right)$ erreurs de *leave-one-out*.

Comme $d = 20$, $S = 20$ et que N prend ses valeurs en fonction du problème de modélisation (voir tableau 7.2), nous avons dans notre cas

7. http ://java.sun.com/
8. http ://www.jfree.org/jfreechart/

dû calculer 4 899 378 estimations d'erreur de *leave-one-out* pour obtenir les résultats du tableau 7.3. Dans le cas des familles d'hypothèses linéaires, les erreurs de *leave-one-out* s'obtiennent rapidement via la méthode PRESS (voir l'annexe B.2 à la page 171).

Nous proposons ici d'utiliser l'algorithme exploreMaxμ_g (voir section 5.3 à la page 107) afin d'accélérer la procédure SFS lorsque des familles d'hypothèses locales (voir section 2.6.2.2 à la page 37) sont utilisées pour calculer les erreurs de *leave-one-out*. Dans notre cas, c'est la famille des hypothèses *lazy-learning* qui est utilisée.

Considérons – à titre d'exemple – le problème d'estimation du signal BIS cinq minutes après la modification du niveau de propofol. Pour cette tâche, 1148 échantillons sont disponibles (voir tableau 7.2) et l'algorithme SFS classique utilise donc $1\,148 \cdot (20\,(20+1)\,/2-1) = 239\,932$ erreurs de *leave-one-out* pour trouver le sous-ensemble optimal. Dans le cas des modèles non-linéaires, ce nombre d'estimations d'erreur de *leave-one-out* devient vite un obstacle pour l'utilisation de l'algorithme SFS.

Nous allons donc proposer une modification de SFS basée sur exploreMaxμ_g afin d'accélérer la procédure de sélection du sous-ensemble de variables optimales en réduisant le nombre d'erreurs de *leave-one-out* nécessaires [9].

A la première étape, plutôt que d'utiliser l'entièreté des $N \times d$ ($= 22\,960$) erreurs de *leave-one-out* pour trouver la variable optimale, nous fixons à $N_{max} \times d$ le nombre de tests maximal disponible où $N_{max} < N$. Il s'agit donc d'un problème de type *selecting the best* (voir section 3.2 à la page 43) où afin de trouver l'optimum, les $N_{max} \times d$ tests disponibles doivent être répartis entre les d alternatives. A la fin de la procédure, n_k représente le nombre de tests réalisés sur la variable k et son erreur de généralisation s'estime comme suit (voir équation (2.15) à la page 34) :

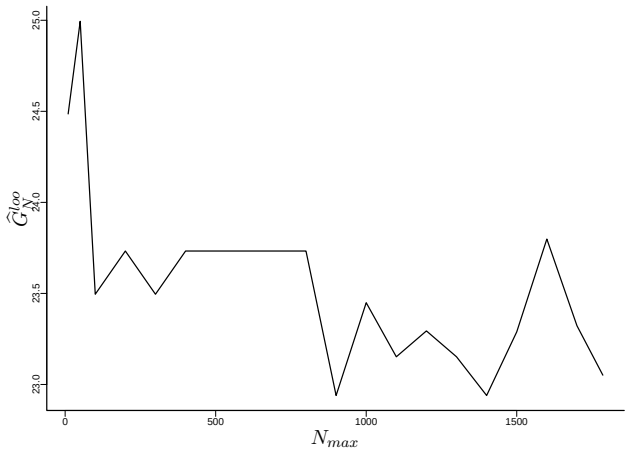
$$\widehat{G}^{loo}_{n_k}(\mathcal{M} = \{k\}) = \frac{1}{n_k} \sum_{i=1}^{n_k} e^i_{loo}.$$

Chaque variable possède donc une estimation de G_{n_k} et la variable avec la meilleure estimation de l'erreur de généralisation est sélectionnée.

La sélection de la deuxième variable optimale se réalise de la même manière à l'exception qu'ici l'algorithme exploreMaxμ_g ne peut utiliser que $N_{max} \times (d-1)$ tests pour choisir parmi $d-1$ alternatives celle qui (combinée avec la variable sélectionnée à l'étape précédente) est optimale.

La procédure se poursuit ainsi de suite sur $d-1$ étapes et c'est finalement l'ensemble de variables ayant la plus petite erreur de généralisation qui est sélectionné.

9. Il est à noter que Maron et Moore ont déjà proposé dans [59] d'utiliser les méthodes de *racing* (voir section 3.2.4 à la page 51) dans le même contexte.



N_{max}	10	50	100	200	300	400	500
$\widehat{G}_N^{loo}$	24.4	24.9	23.4	23.7	23.4	23.7	23.7
p-val	< 0.01	< 0.01	0.32	0.13	0.32	0.13	0.13
N_{max}	600	700	800	900	1000	1100	1200
$\widehat{G}_N^{loo}$	23.7	23.7	23.7	22.9	23.4	23.1	23.2
p-val	0.13	0.13	0.13	0.78	0.36	0.81	0.54
N_{max}	1300	1400	1500	1600	1700	1786	
$\widehat{G}_N^{loo}$	23.1	22.9	23.2	23.7	23.3	23.0	
p-val	0.81	0.78	0.59	0.10	0.54	1	

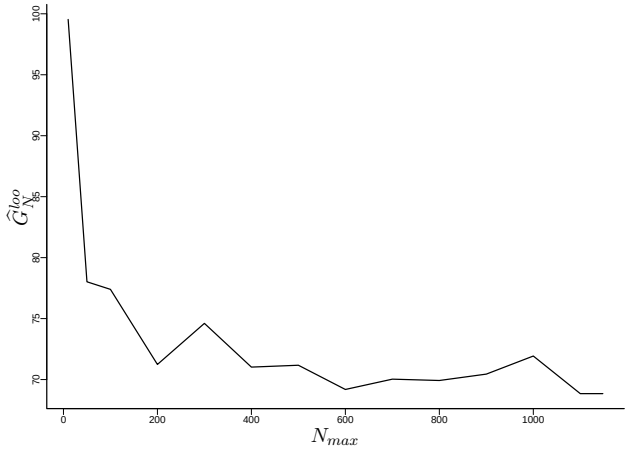
FIGURE 7.10 – Résultat de la sélection de variable sur l'ensemble $D_N^{\Delta t=30}$.

Nous avons testé la procédure sur trois ensembles différents (voir tableau 7.2) : $D_N^{\Delta t=30}$, $D_N^{\Delta t=300}$ et $D_N^{\Delta t=600}$. Sur chaque ensemble, la procédure a été utilisée avec une valeur différente de N_{max}. Ceci permet d'étudier l'évolution des performances de la procédure lorsque N_{max} change.

Dans le premier cas, N_{max} prend les dix-neuf valeurs de $\{10, 50, 100, 200, 300, \ldots, 1600, 1700\}$, dans le deuxième cas, N_{max} prend les treize valeurs de $\{10, 50, 100, 200, 300, \ldots, 1000, 1100\}$ et dans le dernier cas, N_{max} prend les dix valeurs de $\{10, 50, 100, 200, 300, \ldots, 700, 800\}$.

Pour chacun des trois problèmes, nous avons obtenu une série de sous-ensembles de variables correspondant aux variables sélectionnées en utilisant différentes valeurs de N_{max}. Pour comparer les performances des ensembles de variables sélectionnées, les quantités $\widehat{G}_N^{loo}$ ont été calculées sur l'ensemble des N exemples de D_N et les résultats – pour les trois problèmes – sont repris dans les figures 7.10, 7.11 et 7.12.

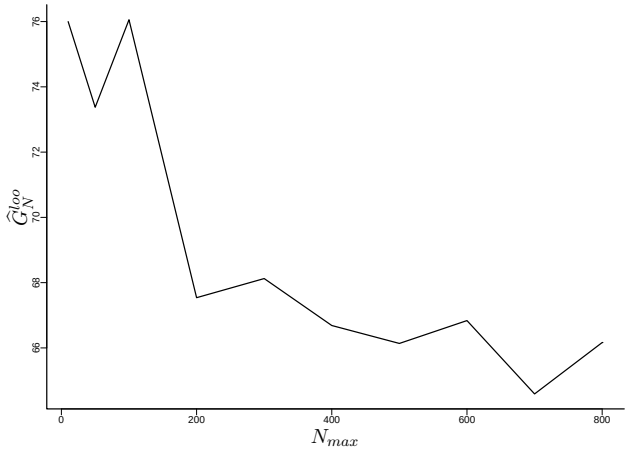
Les graphiques des parties supérieures des figures indiquent l'évolution de $\widehat{G}_N^{loo}$ des sous-ensembles de variables sélectionnées par exploreMaxμ_g lorsque



N_{max}	10	50	100	200	300	400	500
$\widehat{G}_N^{loo}$	99.5	78.0	77.3	71.2	74.6	71.0	71.1
p-val	< 0.01	< 0.01	< 0.01	0.09	< 0.01	0.25	0.16

N_{max}	600	700	800	900	1000	1100	1148
$\widehat{G}_N^{loo}$	69.1	70.0	69.9	70.4	71.9	68.8	68.8
p-val	0.7	0.37	0.26	0.39	0.06	1	1

FIGURE 7.11 – Résultat de la sélection de variable sur l'ensemble $D_N^{\Delta t=300}$.



N_{max}	10	50	100	200	300	400
$\widehat{G}_N^{loo}$	75.9	73.3	76.0	67.5	68.1	66.6
p-val	< 0.01	< 0.01	< 0.01	0.50	0.38	0.79

N_{max}	500	600	700	800	801	
$\widehat{G}_N^{loo}$	66.1	66.8	64.5	66.1	66.1	
p-val	0.98	0.74	0.39	1	1	

FIGURE 7.12 – Résultat de la sélection de variable sur l'ensemble $D_N^{\Delta t=600}$.

N_{max} évolue[10]. Les tableaux de la partie inférieure des figures indiquent les différentes valeurs de $\widehat{G}_N^{loo}$ ainsi que les p-valeurs des t-tests jumelés qui comparent les erreurs des sous-ensembles de variables sélectionnées par l'algorithme exploreMaxμ_g avec celles sélectionnées par SFS.

Une faible p-valeur (plus petite que 0.01) signifie qu'il y a une différence statistique significative entre l'erreur de généralisation du sous-ensemble de variables sélectionné en utilisant $N_{max} \times \left(\frac{d(d+1)}{2} - 1\right)$ estimations d'erreurs de *leave-one-out* et le sous-ensemble de variables sélectionné en utilisant toutes les $N \times \left(\frac{d(d+1)}{2} - 1\right)$ estimations d'erreurs de *leave-one-out* disponibles.

Les résultats montrent très rapidement – déjà à $N_{max} = 200$ – que les sous-ensembles de variables sélectionnées par exploreMaxμ_g obtiennent une performance de généralisation statistiquement proche de celles sélectionnées par SFS. Cette technique permet donc d'accélérer la procédure de sélection de variables sans détériorer significativement les résultats.

7.5 Conclusions

Dans cette section, nous avons pu appliquer une techniques d'apprentissage présentées dans ce travail sur des données réelles provenant du milieu médical.

Notre première contribution est le développement d'un système d'aide à la décision permettant d'estimer le niveau de la profondeur d'inconscience du patient quelques minutes après une modification de concentration de la drogue hypnotique. Ce système a été testé sur un petit groupe de patients mais il est prévu de réaliser – à court terme – un protocole d'étude à plus grande échelle afin d'évaluer le gain d'un tel système.

Il est important, dans les problèmes d'apprentissage artificiel, d'appliquer une procédure de sélection de variables afin de sélectionner celles qui apportent le plus d'informations. Notre deuxième contribution est l'utilisation de l'algorithme exploreMaxμ_g dans une procédure de sélection de variables séquentielle de type SFS afin de réduire le nombre de tests nécessaires. Il est apparu que les résultats sont très rapidement équivalents à ceux obtenus par SFS lorsqu'il utilise toutes les erreurs de *leave-one-out*.

Cette étude s'est focalisée sur l'impact du propofol sur le BIS. La méthode proposée dans ce chapitre peut cependant être appliquée sur d'autres problématiques médicales. Il est prévu – à court terme – d'appliquer ces méthodes afin de prédire le niveau de pression sanguine et le rythme cardiaque du patient après une modification du niveau d'analgésique.

10. Nous avons également donné – à titre indicatif – la valeur de $\widehat{G}_N^{loo}$ lorsque tous les échantillons sont utilisés pour choisir le sous-ensemble de variables. Il est à noter que ce cas correspond au résultat donné par l'algorithme SFS.

Chapitre 8

Conclusions

Cette thèse a proposé de nouveaux algorithmes pour résoudre les deux problèmes d'optimisation stochastique suivants : le problème de la sélection de l'optimum avec un budget de tests limité et le *multi-armed bandit problem*. Tous les algorithmes proposés sont basés sur une quantité appelée « *gain espéré d'une action gloutonne* » (noté μ_g). Une attention particulière a également été portée à la résolution des problèmes de sélection de modèles en apprentissage artificiel afin de développer un système d'aide à la décision en anesthésie.

1. Le gain espéré d'une action gloutonne.

   (a) L'action gloutonne remplit un rôle très important dans ces deux types de problème. Dans le problème de sélection de l'optimum avec un budget de tests limité, après la phase dans laquelle les alternatives sont testées, c'est un algorithme de type glouton qui est utilisé pour chercher l'optimum. Dans le problème du bandit, les stratégies semi-uniformes présentent deux modes de fonctionnement bien distincts : soit elles explorent les différentes alternatives, soit elles exploitent – via une action gloutonne – l'alternative qui semble constituer l'optimum.

   (b) Vu que l'action gloutonne est réalisée dans un environnement aléatoire, son résultat est également aléatoire. Nous avons proposé et donné une définition analytique d'une nouvelle quantité appelée « *gain espéré d'une action gloutonne* » et notée μ_g. Cette quantité est définie à partir de la notion de "*probabilité de faire une sélection correcte*" bien connue en simulation Monte Carlo. Pour un nombre déterminé de tests sur les différentes alternatives, cette quantité μ_g correspond à la valeur moyenne du gain, obtenue après une sélection de type glouton.

   (c) Une étude a été réalisée concernant l'évolution de la valeur de μ_g lorsque le nombre de tests sur les alternatives change. Lorsque

plus de deux alternatives sont considérées, l'étude de l'évolution de μ_g a montré que l'exécution d'un test sur l'alternative optimale peut diminuer la valeur de μ_g quand les mauvaises alternatives ont été insuffisamment explorées. Nous avons montré que ce phénomène contre-intuitif ne se produit pas lorsque $K = 2$.

(d) La définition analytique de μ_g suppose que les caractéristiques (μ et σ) des alternatives sont connues, mais en général cette supposition est irréaliste car seules quelques observations de gains des alternatives sont disponibles. Une série d'estimateurs de μ_g ont donc été proposés et comparés expérimentalement.

2. Le problème de sélection de l'optimum.

   (a) Trois algorithmes ont été proposés pour résoudre le problème de sélection de l'optimum : optiK2Estim, *PCSsearch* et exploreMaxμ_g.
   - L'algorithme optiK2Estim est spécifique au cas où seules deux alternatives sont en compétition. Elle sélectionne l'alternative à tester en fonction des estimations des deux variances et du nombre de tests réalisés sur les alternatives.
   - L'algorithme *PCSsearch* teste – à chaque étape – l'alternative augmentant le plus l'estimation de la « probabilité de faire une sélection correcte » .
   - L'algorithme exploreMaxμ_g utilise les considérations faites sur l'évolution de μ_g afin d'éviter les situations où μ_g décroît.

   (b) Des tests ont été réalisés sur des données synthétiques et sur des problèmes de sélection de modèles. Ils ont comparé les algorithmes *PCSsearch* et exploreMaxμ_g avec les algorithmes $\mathcal{S} + \mathcal{R}$ et F-race provenant de l'état de l'art. Les résultats montrent que la stratégie exploreMaxμ_g donne de meilleurs résultats que la stratégie *PCSsearch* et que exploreMaxμ_g est compétitif avec les deux autres stratégies de l'état de l'art.

3. Le multi-armed bandit problem

   (a) Deux algorithmes semi-uniformes ont été proposés pour résoudre le problème du *bandit* : ϵ-PCSgreedy et DP-*greedy*.
   - L'algorithme ϵ-PCSgreedy est une version modifiée d'ϵ-greedy où la phase d'exploration aléatoire est remplacée par une exploration de type *PCSsearch*.
   - L'algorithme DP-*greedy* utilise un cadre de programmation dynamique afin d'étudier l'impact d'une sélection sur le gain espéré d'une action gloutonne.

   (b) Les résultats expérimentaux ont montré que ϵ-PCSgreedy et DP-*greedy* sont compétitifs avec les stratégies faisant partie de l'état de l'art. La stratégie DP-*greedy* a montré une bonne capacité

d'adaptation aux problèmes de sélection en basculant automatiquement dans un mode d'exploitation lorsque suffisamment d'informations ont été collectées sur les alternatives.

4. Un système d'aide à la décision en anesthésie

   (a) Cette thèse a été réalisée dans le cadre d'un projet de type First Europe Objectif 1 de la région Wallonne en collaboration avec le service d'anesthésie de l'Hôpital Erasme. Le but du projet était d'utiliser les archives d'ITB afin de développer, via des méthodes d'apprentissage artificiel, un système d'aide à la décision. Un prototype a été développé et testé durant un nombre restreint d'interventions.

   (b) Ceci a également permis d'appliquer avec succès l'algorithme exploreMaxμ_g sur un problème de sélection de variables construit sur des données réelles. L'algorithme exploreMaxμ_g a pu trouver un sous-ensemble de variables pertinent en un nombre réduit de tests.

8.1 Perspectives futures

Cette section esquisse quelques directions pour la suite des travaux entamés dans cette thèse. Les quatre axes de recherche principaux en sont :

1. Le calcul du gain espéré d'une action gloutonne, noté μ_g.

   (a) Vu l'importance des actions gloutonnes dans les problèmes de sélection de l'optimum et dans les problèmes de *bandit*, il est intéressant d'étudier l'évolution de $\Pr\{S_{[K]}\}$ lors d'une modification du nombre de tests sur les alternatives. L'étude de l'évolution de $\Pr\{S_{[K]}\}$ a permis de définir la stratégie d'exploration "optiK2" spécifique pour le cas où $K = 2$ (voir section 4.2.1). Concernant la configuration plus générale ($K > 2$), nous avons seulement montré de manière *empirique* la complexité de l'évolution de μ_g. Une direction pour la suite des recherches serait l'étude du cas spécifique où trois alternatives ($K = 3$) se trouvent en compétition.

   Dans ce cas (voir théorème 4.1), $\Pr\{S_{[K]}\}$ est la probabilité que $\widehat{\mathbf{r}}_{[K-1]}$ et $\widehat{\mathbf{r}}_{[K-2]}$ soient positifs où $\left(\widehat{\mathbf{r}}_{[K-1]}, \widehat{\mathbf{r}}_{[K-2]}\right)^T$ suit une distribution normale multivariée

   $$\left(\widehat{\mathbf{r}}_{[K-1]}, \widehat{\mathbf{r}}_{[K-2]}\right)^T \sim \mathcal{N}\left[\Gamma, \Sigma\right]$$

   de moyenne $\Gamma = \left(\mu_{[K]} - \mu_{[K-1]}\ ,\ \mu_{[K]} - \mu_{[K-2]}\right)^T$ et de matrice

de covariance

$$\Sigma = \begin{pmatrix} \frac{\sigma^2_{[K]}}{n_{[K]}} + \frac{\sigma^2_{[K-1]}}{n_{[K-1]}} & \frac{\sigma^2_{[K]}}{n_{[K]}} \\ \frac{\sigma^2_{[K]}}{n_{[K]}} & \frac{\sigma^2_{[K]}}{n_{[K]}} + \frac{\sigma^2_{[K-2]}}{n_{[K-2]}} \end{pmatrix}.$$

Tester une alternative modifie la matrice de covariance de $(\widehat{\mathbf{r}}_{[K-1]}, \widehat{\mathbf{r}}_{[K-2]})^T$. Il s'agirait donc d'étudier les situations où tester l'alternative optimale diminue la valeur de $\Pr\{S_{[K]}\}$.

(b) Les caractéristiques des alternatives sont généralement inconnues ce qui mène à l'utilisation de méthodes d'estimation. La section 4.4.5 a comparé expérimentalement différents estimateurs sur sept tâches d'estimation créées de manière synthétique. Les résultats ont montré que la performance des estimateurs dépend de la difficulté du problème de sélection.

Sur les tâches où il est facile de trouver l'alternative optimale ($\Pr\{S_{[K]}\}$ proche de un), l'estimateur naïf ($\widehat{\boldsymbol{\mu}}_g^{max}$) donne de bons résultats malgré son biais. Par contre, sur les tâches où la sélection de l'optimum est plus difficile, c'est l'estimateur par *leave-one-out* ($\widehat{\boldsymbol{\mu}}_g^{loo}$) qui donne les meilleurs résultats. Une piste intéressante est l'estimateur qui utilise un paramètre λ afin de combiner $\widehat{\boldsymbol{\mu}}_g^{max}$ et $\widehat{\boldsymbol{\mu}}_g^{loo}$. Il reste encore à définir une méthode pour fixer λ.

(c) Les quantités μ_g et μ_r correspondent respectivement à l'*espérance* du gain d'une action gloutonne et aléatoire. Or, lorsqu'une action est exécutée, il peut également être intéressant de prendre la variance du gain en considération. L'étude de la *variance* du gain d'une action gloutonne et aléatoire pourrait être envisagée.

2. La sélection de l'optimum avec un budget de tests limité.

   (a) Les performances de l'algorithme exploreMaxμ_g se détériorent lorsqu'il est appliqué sur des problèmes de sélection de modèles (voir section 5.4.2). Ceci est probablement dû à l'évolution de μ_g qui est différente lorsque l'hypothèse de normalité n'est plus vérifiée (voir section 4.1.2). Une version de exploreMaxμ_g spécifique aux problèmes de sélection de modèles devrait être proposée.

   (b) Il serait intéressant de tester les stratégies de bandit (comme UCB1) sur des problèmes de sélection de l'optimum afin de mesurer leur efficacité sur ce type de problèmes.

3. Le problème du *bandit*.

   (a) La quantité μ_g a uniquement été utilisée pour proposer des algorithmes semi-uniformes. Cette quantité doit également pouvoir être utilisée pour proposer d'autres types de stratégies.

(b) La stratégie DP-*greedy* s'est montrée compétitive avec les stratégies semi-uniformes de l'état de l'art. Néanmoins, des tests préliminaires comparant DP-*greedy* avec des stratégies *non* semi-uniformes (UCB1 et Gittins) ont montré que – excepté sur les problèmes faciles ($\Pr\{S_{[K]}\}$ proche de un) – les performances de DP-*greedy* étaient très en deçà de celles des stratégies *non* semi-uniformes. De nouveaux tests doivent confirmer ces premiers résultats.

Mais ceci peut s'expliquer par la nature même des stratégies semi-uniformes qui – à chaque étape l – ont le choix entre seulement deux actions (pure exploitation ou pure exploration) tandis que les autres stratégies peuvent poser des actions beaucoup plus fines en choisissant directement l'alternative à tester. Il peut par exemple être utile de tester la deuxième alternative optimale $[\widehat{\mathbf{K-1}}]$ mais ce type d'action n'est – par définition – pas accessible aux stratégies semi-uniformes.

(c) La stratégie DP-*greedy* utilise μ_g et μ_r afin de donner un score aux stratégies gloutonne et aléatoire pour ensuite exécuter l'action obtenant le meilleur score. Si les gains espérés d'autres types d'action peuvent être définis, il sera alors possible de les inclure dans DP-*greedy*.

4. L'application des méthodes d'apprentissage artificiel en anesthésie.

(a) Seul l'algorithme exploreMaxμ_g, de type *selecting the best*, a été utilisé sur les problèmes d'anesthésie afin de réaliser une sélection de variables. Il serait intéressant d'utiliser des algorithmes de type *bandit* afin de permettre au système d'aide à la décision d'ajuster ses prédictions *durant* l'intervention.

(b) Le logiciel BisPrediction – testé à l'Hôpital Erasme – implémente uniquement un modèle linéaire afin de prédire le niveau de conscience du patient. Ce logiciel pourrait être perfectionné en utilisant des hypothèses non-linéaires de type *lazy-learning* afin d'améliorer la qualité des prédictions.

(c) Dans cette thèse, nous nous sommes concentrés sur l'étude des relations entre le Propofol et le BIS. Il peut également être utile d'utiliser le même type de système d'aide à la décision afin de prédire l'évolution du rythme cardiaque et de la pression artérielle.

(d) Afin de pallier les problèmes de variabilité interindividuelle, nous avons proposé d'utiliser les premiers instants de l'intervention chirurgicale afin de calculer des variables d'initialisation mesurant la sensibilité du patient aux drogues. Trois des huit variables sélectionnées par la procédure SFS sont des variables d'initialisation. Il serait intéressant de poursuivre cette étude en proposant de nouvelles variables d'initialisation.

Annexe A

Résultats expérimentaux de *PCSsearch* et exploreMaxμ_g réalisés sur des données synthétiques

Cette annexe contient les résultats détaillés des expériences réalisées dans la section 5.4.1.2 où l'algorithme F-race a été comparé – sur des données créées de manière synthétique – aux algorithmes *PCSsearch* et exploreMaxμ_g.

Les tableaux A.1, A.2, A.3 et A.4 contiennent respectivement les résultats lorsque α_f vaut 0.05, 0.1, 0.2 et 0.4.

Les expériences sont réalisées sur six problèmes différents où chaque problème est composé de cinq cents tâches. Le nombre de tests maximum H_{max} prend les valeurs : 200, 500, 1000 et 2000. L'algorithme F-race n'utilise pas toujours la totalité des H_{max} tests disponibles. Durant ces expériences, c'est d'abord F-race qui est exécuté et les deux autres algorithmes doivent utiliser le même nombre de tests que F-race.

Dans les tableaux, les trois premières colonnes contiennent respectivement les scores moyens – calculés sur les cinq cents tâches – des différents algorithmes sur les différentes tâches. Les trois colonnes suivantes contiennent les p-valeurs (via des t-tests jumelés) lorsque les paires de stratégies sont comparées. Nous considérons que le score d'une stratégie est significativement différent du score d'une autre stratégie si la p-valeur est inférieure à 0.05.

H_{max} est le nombre de tests maximum autorisé mais – pour chaque tâche – c'est d'abord F-race qui est exécuté et les deux autres algorithmes doivent s'aligner sur le nombre de tests utilisés par F-race. La dernière colonne (H_u) contient la moyenne – sur les cinq-cent tâches – du nombre de tests réellement effectués par les algorithmes.

	$H_{max} = 200$						
	Scores			P-valeurs			
	PCSsearch	exploreMaxμ_g	F-race ($\alpha_f = 0.05$)	*PCSs* − exMμ_g	*PCSs* − F-race	expMμ_g − F-race	H_u
E1	0.6777	0.6845	0.6745	0.2895	0.6648	0.1465	173.6
E2	0.6584	0.6658	0.6378	0.2718	0.0105	4e-04	168.6
E3	0.7089	0.7099	0.6785	0.9055	0.0015	8e-04	196.2
E4	0.6974	0.6961	0.6755	0.8668	0.0214	0.0194	188.7
E5	0.694	0.7134	0.6832	0.0831	0.3349	0.0097	198.6
E6	0.6612	0.7075	0.6683	0	0.4908	6e-04	196.4

	$H_{max} = 500$						
	Scores			P-valeurs			
	PCSsearch	exploreMaxμ_g	F-race ($\alpha_f = 0.05$)	*PCSs* − exMμ_g	*PCSs* − F-race	expMμ_g − F-race	H_u
E1	0.7085	0.7191	0.7122	0.0254	0.525	0.1805	384.7
E2	0.6781	0.6913	0.6788	0.0039	0.9071	0.0161	369.7
E3	0.7496	0.7525	0.7398	0.6475	0.1358	0.0621	465.9
E4	0.7378	0.7403	0.7264	0.7105	0.0912	0.032	444.3
E5	0.7587	0.78	0.7615	0.0076	0.7205	0.0233	491.6
E6	0.7413	0.7889	0.7558	0	0.0624	2e-04	479.2

	$H_{max} = 1000$						
	Scores			P-valeurs			
	PCSsearch	exploreMaxμ_g	F-race ($\alpha_f = 0.05$)	*PCSs* − exMμ_g	*PCSs* − F-race	expMμ_g − F-race	H_u
E1	0.7296	0.7352	0.7334	0.1609	0.3639	0.6097	648.6
E2	0.7046	0.7053	0.6993	0.8524	0.2307	0.1608	624.8
E3	0.7871	0.7819	0.783	0.2965	0.4137	0.8507	860.6
E4	0.7676	0.7728	0.7655	0.355	0.7038	0.1451	811.7
E5	0.8088	0.8337	0.8036	0	0.3634	0	966.2
E6	0.79	0.8239	0.8094	0	0.0059	0.0333	922.5

	$H_{max} = 2000$						
	Scores			P-valeurs			
	PCSsearch	exploreMaxμ_g	F-race ($\alpha_f = 0.05$)	*PCSs* − exMμ_g	*PCSs* − F-race	expMμ_g − F-race	H_u
E1	0.7421	0.7445	0.7457	0.4458	0.2832	0.6331	1028.9
E2	0.7138	0.7127	0.7087	0.7129	0.2213	0.2853	1005.9
E3	0.8076	0.8052	0.8038	0.5105	0.3106	0.7477	1518.6
E4	0.7909	0.7897	0.7872	0.785	0.384	0.542	1419.2
E5	0.8397	0.8641	0.8489	0	0.0941	2e-04	1839.0
E6	0.824	0.8521	0.8341	0	0.1231	0.0025	1728.7

TABLE A.1 –

	$H_{max} = 200$						
	Scores			P-valeurs			
	PCSsearch	exploreMaxμ_g	F-race ($\alpha_f = 0.1$)	*PCSs* − exMμ_g	*PCSs* − F-race	expMμ_g − F-race	H_u
E1	0.669	0.6685	0.6601	0.9192	0.2168	0.2148	146.5
E2	0.6561	0.6594	0.6318	0.5866	0.0019	1e-04	147.5
E3	0.7035	0.709	0.679	0.5241	0.0086	0.001	182.8
E4	0.6978	0.7016	0.6814	0.6119	0.077	0.0245	176.8
E5	0.6907	0.7161	0.6889	0.0244	0.867	0.0163	194.9
E6	0.6732	0.7151	0.6646	4e-04	0.4047	0	189.7

	$H_{max} = 500$						
	Scores			P-valeurs			
	PCSsearch	exploreMaxμ_g	F-race ($\alpha_f = 0.1$)	*PCSs* − exMμ_g	*PCSs* − F-race	expMμ_g − F-race	H_u
E1	0.6931	0.7009	0.694	0.1243	0.877	0.2187	299.4
E2	0.6749	0.6819	0.6732	0.1562	0.7763	0.1249	293.0
E3	0.7477	0.7471	0.7426	0.9295	0.4265	0.5304	407.3
E4	0.7286	0.7367	0.7252	0.2094	0.6267	0.0725	384.4
E5	0.752	0.7776	0.7637	0.0013	0.1501	0.1075	471.1
E6	0.7269	0.7839	0.7505	0	0.0111	2e-04	438.0

	$H_{max} = 1000$						
	Scores			P-valeurs			
	PCSsearch	exploreMaxμ_g	F-race ($\alpha_f = 0.1$)	*PCSs* − exMμ_g	*PCSs* − F-race	expMμ_g − F-race	H_u
E1	0.7095	0.7142	0.707	0.1517	0.597	0.0805	471.5
E2	0.6941	0.6899	0.6858	0.3504	0.0744	0.3789	454.6
E3	0.7738	0.7747	0.7743	0.8635	0.9275	0.9344	704.5
E4	0.7501	0.7642	0.7646	0.0087	0.0132	0.9227	653.8
E5	0.795	0.8122	0.7986	0.0064	0.5997	0.0497	892.9
E6	0.7722	0.8233	0.7981	0	9e-04	2e-04	790.8

	$H_{max} = 2000$						
	Scores			P-valeurs			
	PCSsearch	exploreMaxμ_g	F-race ($\alpha_f = 0.1$)	*PCSs* − exMμ_g	*PCSs* − F-race	expMμ_g − F-race	H_u
E1	0.7154	0.718	0.7146	0.4209	0.8507	0.4132	683.5
E2	0.7005	0.6956	0.6951	0.2136	0.2099	0.8954	670.8
E3	0.7886	0.7898	0.7897	0.7692	0.773	0.984	1155.7
E4	0.7779	0.7786	0.7776	0.8515	0.9518	0.8172	1056.7
E5	0.8222	0.8437	0.8361	2e-04	0.0213	0.182	1575.6
E6	0.8082	0.8301	0.8244	0.0011	0.011	0.3058	1380.6

TABLE A.2 –

	$H_{max} = 200$						
	Scores			P-valeurs			
	PCSsearch	exploreMaxμ_g	F-race ($\alpha_f = 0.2$)	*PCSs* − exMμ_g	*PCSs* − F-race	expMμ_g − F-race	H_u
E1	0.6565	0.663	0.652	0.3634	0.5662	0.0846	109.6
E2	0.6363	0.6343	0.6243	0.7258	0.0449	0.0723	109.4
E3	0.6913	0.691	0.6712	0.975	0.0335	0.0343	154.1
E4	0.6754	0.6776	0.6775	0.7651	0.8217	0.9892	144.7
E5	0.6804	0.6935	0.6836	0.2299	0.7453	0.3657	182.4
E6	0.6633	0.6923	0.6749	0.0095	0.2808	0.1024	172.4

	$H_{max} = 500$						
	Scores			P-valeurs			
	PCSsearch	exploreMaxμ_g	F-race ($\alpha_f = 0.2$)	*PCSs* − exMμ_g	*PCSs* − F-race	expMμ_g − F-race	H_u
E1	0.6617	0.6779	0.6789	0.0139	0.0182	0.8461	186.8
E2	0.6446	0.6533	0.6426	0.1187	0.7142	0.0478	178.9
E3	0.7166	0.7218	0.7258	0.3964	0.1974	0.5768	296.6
E4	0.7002	0.718	0.7121	0.013	0.1027	0.3809	266.6
E5	0.7417	0.7474	0.7484	0.4701	0.437	0.9123	387.5
E6	0.7153	0.7557	0.7369	0	0.022	0.0299	354.0

	$H_{max} = 1000$						
	Scores			P-valeurs			
	PCSsearch	exploreMaxμ_g	F-race ($\alpha_f = 0.2$)	*PCSs* − exMμ_g	*PCSs* − F-race	expMμ_g − F-race	H_u
E1	0.6739	0.6898	0.6881	0.0123	0.0284	0.6956	249.6
E2	0.6521	0.6565	0.6476	0.3983	0.3688	0.0733	242.3
E3	0.7287	0.7352	0.7413	0.2554	0.0404	0.304	454.5
E4	0.7106	0.7312	0.7298	9e-04	0.0027	0.7967	399.0
E5	0.7704	0.7729	0.7797	0.7128	0.235	0.3489	638.1
E6	0.7293	0.7721	0.767	0	0	0.5112	571.5

	$H_{max} = 2000$						
	Scores			P-valeurs			
	PCSsearch	exploreMaxμ_g	F-race ($\alpha_f = 0.2$)	*PCSs* − exMμ_g	*PCSs* − F-race	expMμ_g − F-race	H_u
E1	0.6743	0.6874	0.6908	0.0366	0.0095	0.4496	306.9
E2	0.6564	0.6578	0.653	0.8067	0.4957	0.3437	313.1
E3	0.7371	0.7444	0.7491	0.1577	0.051	0.4147	663.7
E4	0.7163	0.7317	0.7351	0.0125	0.0045	0.5613	553.7
E5	0.7791	0.7898	0.7957	0.1145	0.0282	0.3695	962.2
E6	0.7466	0.781	0.7808	0	0	0.9857	851.1

TABLE A.3 –

	$H_{max} = 200$						
	Scores			P-valeurs			
	PCSsearch	exploreMaxμ_g	F-race ($\alpha_f = 0.4$)	*PCSs* − exMμ_g	*PCSs* − F-race	expMμ_g − F-race	H_u
E1	0.6187	0.6254	0.6328	0.3304	0.0646	0.2609	48.0
E2	0.6041	0.613	0.6144	0.1723	0.1387	0.8342	50.1
E3	0.6582	0.6594	0.6571	0.8727	0.9043	0.7784	91.5
E4	0.647	0.6452	0.6489	0.8277	0.8192	0.6461	80.2
E5	0.6456	0.6636	0.659	0.1161	0.223	0.6691	132.7
E6	0.6392	0.673	0.656	0.0023	0.1067	0.1277	124.5

	$H_{max} = 500$						
	Scores			P-valeurs			
	PCSsearch	exploreMaxμ_g	F-race ($\alpha_f = 0.4$)	*PCSs* − exMμ_g	*PCSs* − F-race	expMμ_g − F-race	H_u
E1	0.6182	0.6263	0.6405	0.2473	0.0029	0.0479	56.5
E2	0.6051	0.6174	0.6176	0.0683	0.068	0.9749	58.9
E3	0.6691	0.6654	0.679	0.6115	0.2054	0.0894	129.3
E4	0.6518	0.6628	0.6558	0.1503	0.6244	0.3501	105.0
E5	0.6798	0.6919	0.6948	0.2075	0.1287	0.7464	201.2
E6	0.6502	0.6763	0.6894	0.0091	0	0.1647	180.8

	$H_{max} = 1000$						
	Scores			P-valeurs			
	PCSsearch	exploreMaxμ_g	F-race ($\alpha_f = 0.4$)	*PCSs* − exMμ_g	*PCSs* − F-race	expMμ_g − F-race	H_u
E1	0.6199	0.6387	0.6404	0.0076	0.0054	0.7931	58.6
E2	0.6057	0.6098	0.6181	0.5251	0.0694	0.172	62.9
E3	0.6678	0.6732	0.6854	0.4613	0.0239	0.1259	152.4
E4	0.6484	0.6704	0.6585	0.004	0.2205	0.1206	120.1
E5	0.682	0.6995	0.6984	0.0592	0.0678	0.8954	252.0
E6	0.6461	0.685	0.6972	1e-04	0	0.1918	215.3

	$H_{max} = 2000$						
	Scores			P-valeurs			
	PCSsearch	exploreMaxμ_g	F-race ($\alpha_f = 0.4$)	*PCSs* − exMμ_g	*PCSs* − F-race	expMμ_g − F-race	H_u
E1	0.6199	0.6356	0.6404	0.0145	0.0054	0.4312	58.7
E2	0.6057	0.6179	0.6181	0.0533	0.0694	0.9725	64.9
E3	0.6695	0.6792	0.6854	0.1332	0.0383	0.377	167.9
E4	0.6542	0.6589	0.6589	0.5114	0.5735	0.994	133.1
E5	0.6776	0.7053	0.6997	0.0081	0.0303	0.523	286.2
E6	0.6544	0.681	0.6998	0.0061	0	0.0429	244.8

TABLE A.4 –

Annexe B

Les hypothèses linéaires

La régression linéaire [64] est une technique classique qui construit une hypothèse sur les données D_N lorsque l'ensemble Λ ne contient que des hypothèses appartenant à la famille des hypothèses linéaires. Une hypothèse appartient à la famille des hypothèses linéaires si elle a la forme suivante

$$y = x^T a + w \;\Leftrightarrow$$

$$y = [1, x_1, x_2, \ldots, x_d] \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_d \end{bmatrix} + w \tag{B.1}$$

où a est un vecteur contenant les coefficients de l'hypothèse linéaire et w est l'erreur résiduelle.

Cette annexe présente le *least-squares* et la *PRESS* (*Prediction Sum of Squares*). Le *least-squares* est une méthode permettant d'appliquer une identification paramétrique via le principe ERM (voir équation (2.5) à la page 18) sur des classes d'hypothèses appartenant à la famille des hypothèses linéaires. La *PRESS* permet d'obtenir très rapidement, dans le cas d'hypothèses linéaires, une estimation de l'erreur de généralisation G_N (voir équation (2.8) à la page 28) de la classe λ sans devoir réaliser toute la procédure de *cross-validation* (voir section (2.4.3) à la page 31).

B.1 Le least-squares

Appliquer le principe ERM sur un ensemble d'hypothèses appartenant à la famille des hypothèses linéaires consiste à trouver, sur base de D_N, les valeurs optimales des paramètres dans le vecteur a qui minimisent le risque empirique (voir équation (2.4) à la page 18). Le vecteur a optimal est noté

$\widehat{a}$. L'ensemble D_N est composé de N éléments, la formule (B.1) peut donc se réécrire comme suit :

$$Y = Xa + W \Leftrightarrow$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1d} \\ 1 & x_{21} & x_{22} & \dots & x_{2d} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{N1} & x_{N2} & \dots & x_{Nd} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_d \end{bmatrix} + \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix}.$$

Sur base de l'équation (2.4) (à la page 18) et lorsque la fonction de coût $\mathcal{C}$ est quadratique (voir équation (2.6) à la page 19), le risque empirique pour un modèle linéaire peut se réécrire sous forme matricielle de la manière suivante :

$$(Y - Xa)^T (Y - Xa) = R_{emp}(a) \tag{B.2}$$

et l'identification paramétrique cherche le vecteur des paramètres $\widehat{a}$ qui minimise $R_{emp}(a)$, c'est-à-dire

$$\frac{\partial}{\partial a} R_{emp}(a) = 0 \Leftrightarrow$$

$$\frac{\partial}{\partial a} (y - Xa)^T (y - Xa) = 0.$$

Voici les différentes étapes pour trouver les paramètres $\widehat{a}$ optimaux.

$$\begin{aligned} \frac{\partial}{\partial a_i} R_{emp}(a) &= \frac{\partial}{\partial a_i} \left(y^T - a^T X^T\right) (y - Xa) \\ &= 2 \left(a^T X^T - y^T\right) x_i \end{aligned}$$

où x_i est la $i^{ième}$ colonne de$X$

$$\begin{aligned} \left(\frac{\partial}{\partial a} R_{emp}(a)\right)^T &\equiv \left[\frac{\partial}{\partial a_1} R_{emp}(a), \frac{\partial}{\partial a_2} R_{emp}(a), \dots, \frac{\partial}{\partial a_d} R_{emp}(a)\right] \\ &= 2 \left(a^T X^T - y^T\right) X. \end{aligned}$$

On cherche la valeur pour laquelle la dérivée de l'erreur empirique en fonction de a est nulle :

$$
\begin{aligned}
\left(\frac{\partial}{\partial a} R_{emp}(a)\right)^T &= 0 \\
2\left(a^T X^T - y^T\right) X &= 0 \\
a^T X^T - y^T &= 0 \\
Xa - y &= 0 \\
Xa &= y \\
X^T X a &= X^T y \\
\left(X^T X\right)^{-1} X^T X a &= \left(X^T X\right)^{-1} X^T y \\
a &= \left(X^T X\right)^{-1} X^T y.
\end{aligned}
$$

$\widehat{a} = \left(X^T X\right)^{-1} X^T y$ permet de calculer les paramètres optimaux du modèle linéaire. Cette méthode est appelée le *least-squares*.

B.2 La PRESS

Le *leave-one-out cross-validation* (voir section (2.4.3) à la page 31) permet de réaliser une estimation de l'erreur de généralisation (voir équation (2.8) à la page 28) avec un faible biais, mais il est difficile de l'utiliser car il demande de réaliser N identifications paramétriques et N prévisions. La PRESS [64] (*Prediction Sum of Squares*) est une technique qui permet, dans le cas des hypothèses linéaires, de calculer rapidement une estimation de l'erreur de généralisation.

Soit

$$
e_{loo}^j = \left(y_j - x_j^T \widehat{a}^{(j)}\right)^2
$$

l'erreur de *leave-one-out* calculée sur l'échantillon z_j où $\widehat{a}^{(j)}$ correspond aux paramètres d'une hypothèse linéaire construite sur l'ensemble D_N duquel l'échantillon z_j a été retiré.

Pour calculer l'estimation de l'erreur de généralisation par *leave-one-out cross-validation* il faut calculer tous les éléments de l'ensemble $\left\{e_{loo}^1, \ldots, e_{loo}^N\right\}$, ce qui nécessite de faire N identifications paramétriques et N prévisions. Soit H, une matrice appelée *matrice chapeau*

$$
H = X\left(X^T X\right)^{-1} X^T.
$$

Cette matrice chapeau se calcule facilement lors de l'identification paramétrique à partir des least-squares. Le vecteur e tel que $e_j = y_j - x_j^T \widehat{a}$ est ensuite calculé. Dans la PRESS, e_{loo}^j est calculé comme suit :

$$
e_{loo}^j = \left(\frac{e_j}{1 - H_{jj}}\right)^2
$$

où H_{jj} est le $j^{ième}$ élément de la diagonale de H.

L'estimation de l'erreur de généralisation d'une hypothèse linéaire se calcule finalement comme suit :

$$\widehat{G}_N^{press} = \frac{1}{N} \sum_{j=1}^{N} \left(\frac{y_j - x_j^T \widehat{a}}{1 - H_{jj}} \right)^2 .$$

Pour réaliser la PRESS, N prévisions auront été réalisées mais un seul apprentissage.

Annexe C

Définition analytique du regret cumulé

Cette section a pour but de démontrer l'équivalence du *regret cumulé d'une stratégie π sur H étapes* dans les trois définitions suivantes

$$\rho_\pi^H = H \cdot \mu_{[K]} - \sum_{k=1}^{K} \mathbb{E}\big[\mathbf{n}_k^H\big] \cdot \mu_k \tag{C.1}$$

$$= \sum_{k=1}^{K} \mathbb{E}\big[\mathbf{n}_k^H\big] \cdot \Delta_k \tag{C.2}$$

$$= \sum_{l=1}^{H} \delta_\pi^l. \tag{C.3}$$

Nous allons commencer par montrer que $(C.1)$ est équivalent à $(C.2)$:

$$\begin{aligned}
\rho_\pi^H &= H \cdot \mu_{[K]} - \sum_{k=1}^{K} \mathbb{E}\big[\mathbf{n}_k^H\big] \cdot \mu_k \\
&= \sum_{k=1}^{K} \mathbb{E}\big[\mathbf{n}_k^H\big] \cdot \mu_{[K]} - \mathbb{E}\big[\mathbf{n}_k^H\big] \cdot \mu_k \qquad \big(\ \textit{Car } H = \textstyle\sum_{k=1}^{K} I\!E[\mathbf{n}_k^H]\big) \\
&= \sum_{k=1}^{K} \mathbb{E}\big[\mathbf{n}_k^H\big] \cdot \Delta_k.
\end{aligned}$$

Nous allons maintenant montrer que $(C.3)$ est équivalent à $(C.1)$:

$$\begin{aligned}
\rho_\pi^H &= \sum_{l=1}^{H} \delta_\pi^l \\
&= \sum_{l=1}^{H} \mu_{[K]} - \mathbb{E}\big[\mu_{\widehat{\mathbf{k}}} \mid Z_\#\big] \\
&= H \cdot \mu_{[K]} - \sum_{l=1}^{H} \mathbb{E}\big[\mu_{\widehat{\mathbf{k}}} \mid Z_\#\big] \\
&= H \cdot \mu_{[K]} - \sum_{l=1}^{H} \sum_{k=1}^{K} \mathcal{P}_k^l \cdot \mu_k \\
&= H \cdot \mu_{[K]} - \sum_{k=1}^{K} \sum_{l=1}^{H} \mathcal{P}_k^l \cdot \mu_k \\
&= H \cdot \mu_{[K]} - \sum_{k=1}^{K} \mathbb{E}\big[\mathbf{n}_k^H\big] \cdot \mu_k
\end{aligned}$$

où $\mathcal{P}_k^l$ est la probabilité que l'alternative k soit choisie par la stratégie π à l'étape l. Dans le cas d'algorithmes de sélection aléatoire de type uniformes, la valeur de $\mathcal{P}_k^l$ est la même pour chaque alternative et vaut $1/K$. Dans la section 3.1, nous avons vu que dans le cas d'algorithmes gloutons sélectionnant l'alternative avec la plus grande moyenne arithmétique, la valeur de $\mathcal{P}_k^l$ équivaut à $\Pr\{S_k\}$.

Annexe D

La programmation dynamique

Cette annexe a pour but de rappeler les notions de programmation dynamique nécessaires à la compréhension de la section 3.3.5 et du chapitre 6. La programmation dynamique [12] traite des problèmes à temps discret où, à chaque instant v, une décision doit être prise. Une stratégie est une fonction qui, pour chaque état $s \in S$, renvoie une action $u \in U$ où U est un ensemble qui contient toutes les actions possibles. Si à l'état $s^v = i$ l'action u est choisie alors on a une *probabilité de transition* $p_{ij}(u)$ que le prochain état soit j où $i \in S$ et $j \in S$. A l'instant v, une transition de l'état i vers l'état j à la suite d'une action u produit un gain $\alpha^v g(i,u,j)$ où g est une fonction de gain et $0 < \alpha \leq 1$ est un facteur de réduction. Ce facteur de réduction a pour conséquence d'accorder plus d'importance aux gains immédiats.

Nous considérons un problème à horizon fini à V étapes. Dans un problème de programmation dynamique à V étapes, la somme des gains futurs espérés d'une stratégie $\widetilde{\pi}$ qui démarre à l'étape i est

$$J_V^{\tilde{\pi}}(i) = \mathbb{E}\left[\alpha^V \cdot G\left(s^V\right) + \sum_{v=0}^{V-1} \alpha^v \cdot g\left(s^v, u, s^{v+1}\right) \middle| s_0 = i \right]$$

où $\alpha^V \cdot G\left(s^V\right)$ est le gain dans l'état final s^V. Nous définissons

$$J_V^*(i) = \max_{\widetilde{\pi}} J_V^{\widetilde{\pi}}(i)$$

comme la somme des gains maximum pouvant être obtenue sur V étapes si l'état initial est i. Cette fonction de gain optimal J_V^* satisfait l'équation récursive de Bellman

$$J_V^*(i) = \max_{u \in U} \sum_{j \in S} p_{ij}(u) \cdot \left(g(i,u,j) + \alpha J_{V-1}^*(j) \right)$$

où

$$J_0^*(i) = G(i)$$

et la décision optimale u^* est définie comme suit

$$u^* = \arg\max_{u \in U} \sum_{j \in S} p_{ij}(u) \cdot \left(g(i,u,j) + \alpha J_{V-1}^*(j) \right).$$

Annexe E

L'inégalité de Hoeffding

Soit

$$\mathbf{X}_1, \ldots, \mathbf{X}_n$$

une série de variables aléatoires indépendantes et bornées telles que

$$\forall i \in [1, n] \ , \quad \Pr\{\mathbf{X}_i \in [a_i, b_i]\} = 1.$$

Si $\mathbf{S} = \sum_{i=1}^{n} \mathbf{X}_i$ est la somme de ces variables aléatoires et si ϵ est un nombre positif, alors l'inégalité de Hoeffding [46] est une borne supérieure de la probabilité que l'écart entre $\mathbf{S}$ et $\mathbb{E}[\mathbf{S}]$ soit plus grand que $n \cdot \epsilon$

$$\Pr\{|\mathbf{S} - \mathbb{E}[\mathbf{S}]| \geq n \cdot \epsilon\} \leq 2 \cdot \exp\left(-\frac{2 \cdot n^2 \cdot \epsilon^2}{\sum_{n}^{i=1} (b_i - a_i)^2}\right). \tag{E.1}$$

Il est à noter que si les variables $\mathbf{X}_1, \ldots, \mathbf{X}_n$ sont des variables indépendantes et identiquement distribuées de moyenne μ et si elles sont bornées dans l'intervalle [0,B] alors l'équation (E.1) peut se réécrire comme suit

$$\Pr\{|\widehat{\boldsymbol{\mu}} - \mu| \geq \epsilon\} \leq 2 \cdot \exp\left(-\frac{2 \cdot n \cdot \epsilon^2}{B^2}\right)$$

où $\widehat{\boldsymbol{\mu}} = \mathbf{S}/n$.

Annexe F

La loi de distribution khi-carré

Considérons v variables aléatoires indépendantes qui suivent des lois de distribution normale, de moyenne nulle et d'écart-type un :

$$\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X_v}.$$

La variable aléatoire

$$\mathbf{X} = \sum_{i=1}^{v} \mathbf{X_i}^2$$

suit une loi de distribution *khi-carrée à v degrés de liberté* (χ^2_v), où le nombre de degrés de liberté v représente le nombre de variables aléatoires qui interviennent dans la définition de $\mathbf{X}$.

L'espérance et la variance d'une variable $\mathbf{X}$ qui suivent la loi de distribution χ^2_v sont définies comme suit :

$$\mathbb{E}[\mathbf{X}] = v \quad , \quad \mathrm{Var}[\mathbf{X}] = 2v.$$

La valeur de la fonction de densité $\Pr\{\mathbf{X} = x\}$ de cette distribution khi-carrée est nulle pour $x < 0$ et pour $x > 0$ la forme de $\Pr\{\mathbf{X} = x\}$ dépend du degré de liberté v (voir figure F.1).

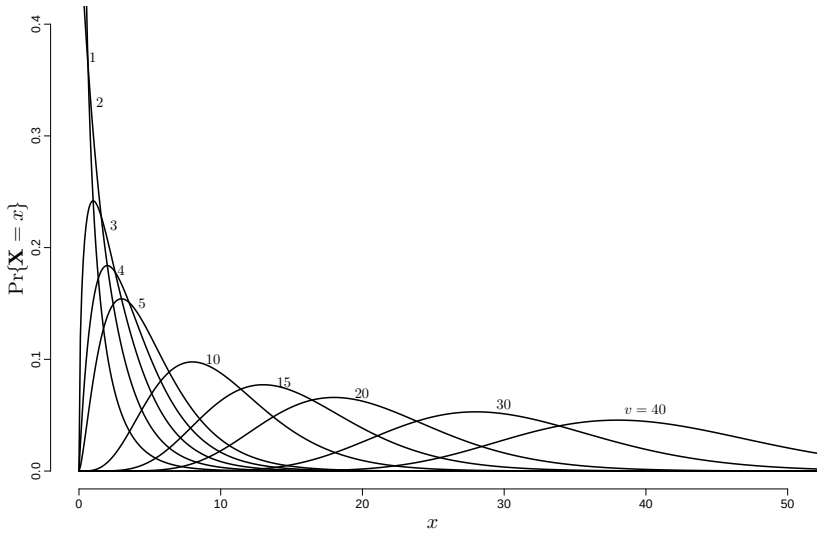


FIGURE F.1 – Fonction de densité de χ^2_v

Annexe G

Le maximum d'un ensemble de variables aléatoires

La quantité $\Pr\{S_{[K]}\}$ est la probabilité que la variable aléatoire $\widehat{\boldsymbol{\mu}}_{[K]}$ soit plus grande que la variable aléatoire $\mathbf{X}_{max} = \max_{k\in\{1K\}/[K]} \widehat{\boldsymbol{\mu}}_k$. Vu son importance, cette section est consacrée à l'étude des propriétés du maximum d'un ensemble de variables aléatoires.

Soit $\mathbf{Y}_1, \mathbf{Y}_2, \mathbf{Y}_3, \ldots, \mathbf{Y}_K$ une série de variables aléatoires. Si ces variables sont indépendantes et identiquement distribuées [1] alors une forme analytique de la distribution de $\mathbf{Y}_{max} = \max_k \mathbf{Y}_k$ peut être obtenue pour certaines distributions de $\mathbf{Y}_1$ [35].

Dire que $\mathbf{Y}_{max}$ est plus petit ou égal à une certaine valeur est équivalent à ce que toutes les valeurs $\mathbf{Y}_1, \mathbf{Y}_2, \mathbf{Y}_3, \ldots, \mathbf{Y}_K$ soient plus petites que y. Ceci implique que si $\mathbf{Y}$ est une variable de même distribution que les variables $\mathbf{Y}_1, \mathbf{Y}_2, \mathbf{Y}_3, \ldots, \mathbf{Y}_K$ alors

$$\Pr\{\mathbf{Y}_{max} \leq y\} = (\Pr\{\mathbf{Y} \leq y\})^K$$

et la fonction de répartition de $\mathbf{Y}_{max}$ est

$$F_{\mathbf{Y}_{max}}(y) = (F_{\mathbf{Y}}(y))^K$$

et la fonction de distribution de $\mathbf{Y}_{max}$ est donc

$$\Pr\{\mathbf{Y}_{max} = y\} = K \cdot \Pr\{\mathbf{Y} = y\} \cdot (F_{\mathbf{Y}}(y))^{K-1}. \tag{G.1}$$

Comme application de (G.1), considérons que $\mathbf{Y}$ suit une loi uniforme dans l'intervalle $[0, L]$. Comme (pour $0 \leq y \leq L$)

$$\Pr\{\mathbf{Y} = y\} = \frac{1}{L} \quad \text{et} \quad F_{\mathbf{Y}}(y) = \frac{y}{L},$$

1. Il est à noter que dans le cas $\widehat{\boldsymbol{\mu}}_1, \widehat{\boldsymbol{\mu}}_2, \ldots, \widehat{\boldsymbol{\mu}}_K$ les variables sont indépendantes et non identiquement distribuées.

la fonction de distribution de $\mathbf{Y}_{max}$ devient

$$\Pr\{\mathbf{Y}_{max} = y\} = \frac{K \cdot y^{K-1}}{L^K}$$

et la moyenne et la variance de $\mathbf{Y}_{max}$ sont respectivement

$$\frac{K \cdot L}{K+1} \quad , \quad \frac{K \cdot L^2}{(K+1)^2(K+2)}.$$

Bibliographie

[1] D. W. Aha and R. L. Bankert. A comparative evaluation of sequential feature selection algorithms. *Artificial Intelligence and Statistics*, 5, 1996.

[2] H. Akaike. Statistical predictor identification. *Ann. Inst. Stat. Math.*, 22 :203–217, 1970.

[3] H. Akaike. A new look at the statistical model identification. 19(6) :716–723, 1974.

[4] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11(1–5) :11–73, 1997.

[5] J.-Y. Audibert, R. Munos, and C. Szepesvári. Use of variance estimation in the multi-armed bandit problem. In *NIPS 2006 Workshop on On-line Trading of Exploration and Exploitation*, 2006.

[6] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2/3) :235–256, 2002.

[7] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino : the adversarial multi-armed bandit problem. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 322–331. IEEE Computer Society Press, Los Alamitos, CA, 1995.

[8] J. M. Bailey, W. M. Haddad, and T. Hayakawa. Closed-loop control in clinical pharmacology : Paradigms, benefits and challenges. In *Proceedings of the 2004 American Control Conference*, pages 2268–2277, 2004.

[9] R.E. Bechhofer. A single-sample multiple decision procedure for ranking means of normal populations with known variances. *Annals of Mathematical Statistics*, 25(1) :16–39, 1954.

[10] R.E. Bechhofer, T.J. Santner, and D.M. Goldsman. *Design and Analysis of Experiments for Statistical Selection, Screening, and Multiple Comparisons*. John Wiley & Sons, 1995.

[11] D. Berry and B. Fristedt. *Bandit problems : Sequential Allocation of Experiments*. Chapman and Hall, 1985.

[12] D. P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. I.* Athena Scientific, 2007.

[13] M. Birattari. *Tuning Metaheuristics : A Machine Learning Perspective*, volume 197 of *Studies in Computational Intelligence*. IOS Press, springer edition, 2005.

[14] M. Birattari and G. Bontempi. The lazy learning toolbox, for use with matlab. Technical Report TR/IRIDIA/99-7, IRIDIA-ULB, Brussels, Belgium, 1999.

[15] M. Birattari, G. Bontempi, and H. Bersini. Lazy learning meets the recursive least-squares algorithm. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *NIPS 11*, pages 375–381, Cambridge, 1999. MIT Press.

[16] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp. A racing algorithm for configuring metaheuristics. In W. B. Langdon, editor, *GECCO 2002*, pages 11–18. Morgan Kaufmann, 2002.

[17] C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[18] G. Bontempi. *Local Learning Techniques for Modeling, Prediction and Control*. PhD thesis, IRIDIA- Université Libre de Bruxelles, 1999.

[19] G. Bontempi, M. Birattari, and H. Bersini. A model selection approach for local learning. *Artificial Intelligence Communications*, 13(1) :41–48, 2000.

[20] G. Bontempi, M. Birattari, and P.E. Meyer. Combining lazy learning, racing and subsampling for effective feature selection. In *Adaptive and Natural Computing Algorithms : Proceedings of the International Conference ICANNGA05*, pages 393–396. Springer Computer Science, 2005.

[21] M. D. Buhmann. *Radial Basis Functions : Theory and Implementations*. Cambridge University Press, 2003.

[22] K. P. Burnham and D. R. Anderson. *Model Selection and Multimodel Inference : A Practical Information Theoretic Approach*. Springer, 2004.

[23] F. Cantraine and E. Coussaert. The first object oriented monitor for intravenous anesthesia. *PubMed - indexed for MEDLINE*, 16 :3–10, 2000.

[24] N. Cesa-Bianchi and P. Fischer. Finite-time regret bounds for the multiarmed bandit problem. In *Proc. 15th International Conf. on Machine Learning*, pages 100–108. Morgan Kaufmann, San Francisco, CA, 1998.

[25] S. E. Chick. Selecting the best system : a decision-theoretic approach. In *WSC '97 : Proceedings of the 29th conference on Winter simulation*, pages 326–333, Washington, DC, USA, 1997. IEEE Computer Society.

[26] W. J. Conover. *Practical Nonparametric Statistics*. John Wiley & Sons, New York, NY, USA, third edition, 1999.

[27] A. Cornuéjols and L. Miclet. *Apprentissage artificiel - Concepts et algorithmes*. Eyrolles, 2002.

[28] L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer Verlag, 1996.

[29] N. R. Draper and H. Smith. *Applied Regression Analysis*. John Wiley and Sons, New York, 1981.

[30] G. Dreyfus, J.-M. Martinez, M. Samuelides, M.B. Gordon, F. Badran, S. Thiria, and L. Hérault. *Réseaux de neurones - Méthodologie et applications*. Eyrolles, 2004.

[31] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. John Wiley and sons, 2001.

[32] E. J. Dudewicz and S. R. Dalal. Allocations of observations in ranking and selection with unequal variances. *Sanhkya*, 37 :28–78, 1975.

[33] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, New York, 1993.

[34] B. Efron and R.J. Tibshirani. Cross-validation and the bootstrap : Estimating the error rate of a prediction rule. Technical Report 176, MAY 1995.

[35] W.J. Ewens and G. Grant. *Statistical Methods in Bioinformatics : An Introduction*. Springer, second edition, 2005.

[36] J.H Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19(1) :1–67, 1991.

[37] C. Furlanello, S. Merler, A. Rizzoli, and C. Chemini. An application of the bootstrap 632+ rule to ecological data. In M. Marinaro and R. Tagliaferri, editors, *Neural Nets WIRN-97*, pages 227–232. MIT Press, 1997.

[38] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1) :1–58, 1992.

[39] A. Gentilini, C. Frei, A.H. Glattfelder, M. Morari, T. Schnider, T.J. Sieber, R. Wymann, and A.M. Zbinden. Multitasked closed-loop control in anesthesia. *IEEE Engineering in Medicine and Biology Magazine*, 20(1) :39–53, January 2001.

[40] A. Genz. Numerical computation of multivariate normal probabilities. *Journal of Computational and Graphical Statistics*, (1) :141–149, 1992.

[41] A. Genz and F. Bretz. Numerical computation of multivariate t-probabilities with application to power calculation of multiple contrasts. *Journal of Statistical Computation and Simulation*, 63 :361–378, 1999.

[42] J. C. Gittins. *Multi-armed Bandit Allocation Indices*. Wiley, 1989.

[43] S.S. Gupta. On some multiple decision (selection and ranking) rules. *Technometrics*, 7(2) :225–245, 1965.

[44] J. Hardwick and Q. Stout. Bandit strategies for ethical sequential allocation. *Computing Science and Statistics*, 23 :421–424, 1991.

[45] A. J. Hayter. Selecting the normal population with the largest mean when the variances are bounded. *Communications in statistics. Theory and methods*, 18(4) :1455–1467, 1989.

[46] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58 :13–30, 1963.

[47] A. Hope and R. Rosipal. Measuring depth of anesthesia using electroencephalogram entropy rates. Technical report, Computing and Information Systems, University of Paisley, 2001.

[48] K. Inoue, S. E. Chick, and C-H Chen. An empirical evaluation of several methods to select the best system. *ACM Transactions on Modeling and Computer Simulation*, 9(4) :381–407, 1999.

[49] W. James and C. Stein. Estimation with quadratic loss. In Jerzy Neyman, editor, *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability*, pages 361–379. University of California Press, 1961.

[50] L. P. Kaelbling, M. L. Littman, and A. P. Moore. Reinforcement learning : A survey. *Journal of Artificial Intelligence Research*, 4 :237–285, 1996.

[51] S. Kim and B. Nelson. A fully sequential procedure for indifference-zone selection in simulation. *ACM Trans. Model. Comput. Simul.*, 11(3) :251–273, 2001.

[52] S. Kim and B. Nelson. Selecting the best system : theory and methods. In *WSC '03 : Proceedings of the 35th conference on Winter simulation*, pages 101–112. Winter Simulation Conference, 2003.

[53] S. Kim and B. Nelson. *Handbooks in Operations Research and Management Science : Simulation*, chapter 17 : Selecting the Best System. Elsevier Science, 2006.

[54] A. J. Kleywegt, A. Shapiro, and T. Homem de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal of Optimization*, 12 :479–502, 2001.

[55] R Kohavi and GH John. Wrappers for feature subset selection. *AIJ*, 97(1-2) :273–324, 1997.

[56] D. Koller and M. Sahami. Toward optimal feature selection. *International Conference on Machine Learning*, pages 284–292, 1996.

[57] R.D. Luce. *Individual Choice Behavior : A Theoretical Analysis*. Wiley, 1959.

[58] P. C. Mahalanobis. On the generalised distance in statistics. In *Proceedings National Institute of Science, India*, volume 2, pages 49–55, April 1936.

[59] O. Maron and A. Moore. The racing algorithm : Model selection for lazy learners. 11(1–5) :193–225, 1997.

[60] N. Meuleau and P. Bourgine. Exploration of multi-state environments : Local measures and back-propagation of uncertainty. *Machine Learning*, 35(2) :117–154, 1999.

[61] C.F. Minto, T.W. Schnider, T.D. Egan, E. Youngs, H.J. Lemmens, P.L. Gambus, V. Billard, J.F. Hoke, K.H. Moore, D.J. Hermann, K.T. Muir, J.W. Mandema, and S.L. Shafer. Influence of age and gender on the pharmacokinetics and pharmacodynamics of remifentanil. *Anesthesiology*, 86(1) :10–23, 1997.

[62] T. Mitchell. *Machine Learning*. McGraw, 1997.

[63] G. Morgan, M. Mikhail, and M. Murray. *Clinical Anesthesiology*. McGrawHill, 4 edition, 2006.

[64] R. H. Myers. *Classical and Modern Regression with Applications*. PWS-KENT, Boston, MA, 2000.

[65] B. Nelson, J. Swann, D. Goldsman, and W. Song. Simple procedures for selecting the best simulated system when the number of alternatives is large. *Oper. Res.*, 49(6) :950–963, 2001.

[66] M. P. Perrone and L. N. Cooper. When networks disagree : Ensemble methods for hybrid neural networks. In R. J. Mammone, editor, *Artificial Neural Networks for Speech and Vision*, pages 126–142. Chapman and Hall, 1993.

[67] S. Pierret, A. Demeulenaere, B. Gouverneur, and Ch. Hirsch. Designing turbomachinery blades with the function approximation concept and the navier-stokes equations. In *8th AIAA/NASA/USAF/ISSMO Symposium on Multi-Disciplinary Optimization*, Long Beach, CA., 2000.

[68] W.B. Powell. *Approximate Dynamic Programming - Solving the Curses of Dimensionality*. Wiley, Princeton, New Jersey, 2007.

[69] M. L. Puterman. *Markov Decision Processes : Discrete Stochastic Dynamic Programming*. Wiley, 1994.

[70] Y. Rinott. On two-stage selection procedures and related probability-inequalities. *Communications in statistics*, 8 :799–811, 1978.

[71] H. Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5) :527–535, 1952.

[72] B. Roy and D. Bouyssou. *Aide Multicritère à la Décision : Méthodes et Cas*. Economica, Paris, 1993.

[73] P. Sandeep, C. Deepayan, and A. Deepak. Multi-armed bandit problems with dependent arms. In *ICML '07 : Proceedings of the 24th international conference on Machine learning*, pages 721–728. ACM, 2007.

[74] T.W. Schnider, C.F. Minto, S.L. Shafer, P.L. Gambus, C. Andresen, D.B. Goodale, and E.J. Youngs. The influence of age on propofol pharmacodynamics. *Anesthesiology*, 90(6) :1502–1516, 1999.

[75] D. W. Scott. *Multivariate Density Estimation : Theory, Practice, and Visualization (Wiley Series in Probability and Statistics)*. Wiley-Interscience, September 1992.

[76] F. Servin. Anesthésie intraveineuse à objectif de concentration. In et Sfar. Elsevier SAS, editor, *Conférences d'actualisation de la Société Française d'Anesthésie et de Réanimation (SFAR)*, pages 35–48, 1998.

[77] F. Servin. Interactions entre hypnotiques et morphiniques : conséquences pour la pratique. In et Sfar. Elsevier SAS, editor, *Conférences d'actualisation de la Société Française d'Anesthésie et de Réanimation (SFAR)*, pages 349–368, 2001.

[78] J.C. Sigl and N.G. Chamoun. An introduction to bispectral analysis for the electroencephalogram. *Clin Monitor*, 10 :392–404, 1994.

[79] I. Smith and P. White. *Total Intravenous Anaesthesia*. Blackwell BMJ Books, 1998.

[80] J.C. Spall. *Introduction to Stochastic Search and Optimization*. John Wiley & Sons, Inc., New York, NY, USA, 2003.

[81] M Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society B*, 36(1) :111–147, 1974.

[82] R.S. Sutton and A.G. Barto. *Reinforcement Learning : An Introduction*. MIT Press, Cambridge, MA, 1998.

[83] Y.L. Tong. *The Multivariate Normal Distribution*. Springer Verlag, 1990.

[84] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, NY, 1995.

[85] V. N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[86] J. Vermorel and M. Mohri. Multi-armed bandit algorithms and empirical evaluation. In *16th European Conference on Machine Learning (ECML05)*, pages 437–448. ecml, 2005.

[87] C. Watkins. *Learning From Delayed Rewards*. PhD thesis, Cambridge University, 1989.

[88] R. R. Wilcox. A table for rinott's selection procedure. *Journal of Quality Technology*, 1984.

[89] F.Y.H. Yeh and M. Gallagher. An empirical study of hoeffding racing for model selection in k-nearest neighbor classification. In *Intelligent Data Engineering And Automated Learning Ideal 2005*, pages 220–227. Springer-verlag Berlin, 2005.

[90] T. Zikov, S. Bibian, G.A. Dumont, M. Huzmezan, and C.R. Ries. Quantifying cortical activity during general anesthesia using wavelet analysis. In Institute of Electrical and Electronics Engineers, editors, *IEEE transactions on biomedical engineering*, pages 617–632, 2006.