N° D'ORDRE 8636

**UNIVERSITE PARIS-SUD XI**
**Faculté des Sciences d'Orsay**

# THÈSE DE DOCTORAT

**SPECIALITE : PHYSIQUE**

***Ecole Doctorale « Sciences et Technologies de l'Information des Télécommunications et des Systèmes »***

Présentée par : Stéphane GAZUT

Sujet :

Conception et mise en œuvre de nouvelles méthodes d'élaboration de plans d'expériences pour l'apprentissage de modèles non linéaires.

Soutenue le 28 mars 2007 devant les membres du jury :

| | |
|---|---|
| Mme Michèle SEBAG | (Présidente du Jury) |
| M Jean-Pierre GAUCHI | (Rapporteur) |
| M Yves GRANDVALET | (Rapporteur) |
| M Jean-Marc MARTINEZ | (Examinateur) |
| M Eric WALTER | (Examinateur) |
| M Gérard DREYFUS | (Directeur de thèse) |

**Université Paris Sud XI – Orsay**

**Ecole doctorale Sciences et Technologies de l'Information des Telecommunications et des Systèmes**

# Conception et mise en oeuvre de nouvelles méthodes d'élaboration de plans d'expériences pour l'apprentissage de modèles non linéaires.

# THÈSE

présentée et soutenue publiquement le 28 mars 2007

pour l'obtention du grade de

**Docteur de l'université Paris XI**

**(spécialité physique)**

par

Stéphane GAZUT

**Composition du jury**

| | |
|---|---|
| *Président :* | Michèle Sebag |
| *Rapporteurs :* | Jean-Pierre Gauchi<br>Yves Grandvalet |
| *Examinateurs :* | Jean-Marc Martinez<br>Eric Walter |
| *Directeur de thèse :* | Gérard Dreyfus |

**Laboratoire d'Etudes Thermiques des Réacteurs — CEA (Saclay)**

# Résumé

Dans le cadre de la simulation numérique, la thèse aborde le problème posé par la construction de modèles simplifiés de codes de calcul généralement complexes et lourds à mettre en oeuvre. La problématique de construction des modèles simplifiés rappelle celle des surfaces de réponse et des plans d'expériences. Dans le cadre de la simulation numérique, les expériences "lourdes" correspondent aux calculs à réaliser. Il est donc important de disposer de méthodes permettant d'optimiser la planification des expériences simulées.

Les méthodes que nous proposons s'appuient essentiellement sur la théorie de l'apprentissage statistique. Les modèles simplifiés sont construits en utilisant la famille des réseaux de neurones et les techniques statistiques d'estimation de l'erreur de généralisation pour contrôler l'apprentissage (leave-one-out, validation croisée, bootstrap). Il est montré que le bootstrap permet de contrôler le sur-apprentissage (compromis Biais-Variance) et d'étendre la notion de levier (mesure de l'influence des expériences sur l'estimation des paramètres), introduite en régression linéaire, aux modèles non linéaires par rapport aux paramètres.

La thèse propose une procédure itérative de planification d'expériences numériques, LDR (acronyme de Learner Disagreement from experiment Resampling). Elle est fondée sur l'apprentissage actif en procédant à la construction de plusieurs modèles simplifiés à partir de répliques bootstrap. Par une approche de type Bagging (Bootstrap Aggregating), la variance des prédictions des modèles est analysée. Les nouvelles simulations sont planifiées de façon à réduire cette variance des prédictions. Cette procédure est comparée aux approches classiques des plans d'expériences optimaux et en particulier avec la D-optimalité.



**Mots-clés:** Apprentissage, réseaux de neurones, bootstrap, bagging, active learning, plans d'expériences, D-optimalité.

# Abstract

This thesis adresses the problem of the construction of surrogate models in numerical simulation. Whenever numerical experiments are costly, the simulation model is complex and difficult to use. It is important then to select the numerical experiments as efficiently as possible in order to minimize their number. In statistics, the selection of experiments is known as optimal experimental design. In the context of numerical simulation where no measurement uncertainty is present, we describe an alternative approach based on statistical learning theory and resampling techniques. The surrogate models are constructed using neural networks and the generalization error is estimated by leave-one-out, cross-validation and bootstrap. It is shown that the bootstrap can control the over-fitting and extend the concept of leverage for non linear in their parameters surrogate models. The thesis describes an iterative method called LDR for Learner Disagreement from experiment Resampling, based on active learning using several surrogate models constructed on bootstrap samples. The method consists in adding new experiments where the predictors constructed from bootstrap samples disagree most. We compare the LDR method with other methods of experimental design such as D-optimal selection.



**Keywords:** Active Learning, Bagging, Bootstrap, Non linear design of experiments, Neural Networks, D-optimality.

# Remerciements

Je tiens à remercier tout d'abord mon directeur de thèse Gérard DREYFUS pour les remarques constructives dont il m'a fait part pendant la thèse mais également pendant la phase de rédaction. Ses remarques ont permis d'améliorer significativement la qualité du manuscrit.

Je tiens à remercier vivement Jean-Marc MARTINEZ qui fut mon professeur en machine learning durant mon DEA et mon encadrant au CEA durant ma thèse. Je le remercie pour la qualité de son encadrement. Alors que certains doctorants se plaignent parfois de l'indisponibilité de leur encadrant, je dois dire que Jean-Marc a su être disponible, être à l'écoute et tirer les meilleures idées des différentes discussions que nous avons pu avoir. Je le remercie également pour son soutien dans mes différentes démarches pour préparer l'après thèse et je ne peux que me rejouir d'avoir la possibilité de travailler avec lui dans le cadre de mes nouvelles fonctions au CEA.

Je voudrais remercier également Jean-Pierre GAUCHI et Yves GRANDVALET d'avoir accepté de rapporter ma thèse. Les remarques et discussions que j'ai pu établir avec eux ont été très interessantes et instructives.

Mes remerciements vont également à Michèle SEBAG, qui a accepté d'être présidente de mon jury, ainsi qu'à Eric WALTER qui fut examinateur durant la soutenance. J'ai eu l'occasion de les rencontrer à différents séminaires de Digiteo Labs et j'aurai certainement l'occasion d'avoir encore d'autres échanges avec eux.

Je tiens à remercier Yacine OUSSAR, maître de conférence et collaborateur de Gérard DREYFUS au laboratoire d'Electronique de l'ESPCI, avec qui j'ai eu l'occasion de travailler. Les discussions que nous avons pu avoir ensemble m'ont apporté beaucoup tant sur le plan scientifique qu'artistique puisque nous partageons la même passion pour la photographie.

Je voudrais également remercier l'ensemble des équipes qui ont travaillé sur le projet NeuroPex. Je pense bien sûr aux membres de la société Netral, Jean-Luc PLOIX, Patrice KIENER et Sébastien ISSANCHOU avec qui j'ai eu le plaisir de travailler lorsque je me suis employé à faire la comparaison entre la D-optimalité et la méthode LDR. Je pense également à Abdelazziz FARAJ de l'IFP qui n'a malheureusement pas pu être présent pour la soutenance, sans oublier bien sûr mes collègues de DP2I au CEA à savoir Jacques VAN-DER-VLIET, Robert QUACH, Annie MASSON et Nicolas LECLER.

Je tiens à remercier vivement les membres du Laboratoire d'Etudes Thermiques des Réacteurs qui m'ont acceuilli durant ma thèse, Anela, Annalisa, Augustin, Edwige, Josiane, Matthieu, Olivier, Patricia, Philippe, Samuel, Stéphane, Stéphanie, Sylvie ainsi que les autres doctorants du laboratoire Coralie, Florian, François, Mickael, Nadia, Thierry et Vincent qui a soutenu la veille de ma soutenance. Les trois années passées parmi eux ont été très agréables grâce à la très bonne ambiance qui rêgne dans le laboratoire. Je tiens à remercier plus particulièrement, d'une part, le groupe SUMO (Supervision, Modélisation, Optimisation) composé de Jean-Marc, Fabrice, Gilles et Michel, et d'autre part, Eric, chef du LETR, Daniel CARUGE chef du Service Fluides numériques Modélisations et Etudes et Jacques SEGRE qui m'ont également soutenu dans mes différentes démarches pour l'après thèse.

Je remercie vivement Karine AURIBAULT et Nicolas GILARDI avec qui je partage mon bureau depuis février et qui m'ont aidé et m'ont déchargé de toute la partie logistique le jour de

la soutenance. Nicolas était en post-doc à l'IFP au côté d'Abdelazziz Faraj et a travaillé sur le projet NeuroPex, nous ne nous étions pas croisé à l'époque car la fin de son post-doc coïncidait avec le début de ma thèse, comme on s'amuse à le dire souvent : le monde est petit ;-)

Je tiens à saluer vivement les membres du Plan d'Action Neuronal (PAN) au CEA qui se propose d'organiser une fois par trimestre des séminaires sur les problématiques liées à l'apprentissage statistique et plus particulièrement Michaël AUPETIT, ainsi que David MERCIER, Laurence CORNEZ et Jean-Denis MULLER avec qui je travaille au LETS.

J'ai été intégré au laboratoire LETS (Laboratoire d'Electronique et de Traitement du Signal) en novembre 2006 et je voudrais saluer les membres de ce laboratoire que je n'ai pas encore cités : Anthony, Baptiste, Bernard, Bertrand Monfort, Bertrand Rivet, Eric, François, Gwenolé, Jean, Laurent, Tabéa, Thi, Thierry, Thomas et Virginie.

Je tiens à remercier vivement les membres de ma famille qui m'ont toujours encouragé et soutenu et plus particulièrement Marc et Zaïda (mes parents), Vincent et Ingrid (mon frère et ma belle soeur), Sophie et Thomas (ma soeur et mon beau frère). Je remercie Thomas pour ses remarques concernant quelques parties du manuscrit. J'ai une pensée toute particulière pour Sarah, le premier enfant de Sophie et Thomas, qui est née la veille de ma soutenance. J'ai également une pensée affective pour mes deux autres neveux Nicolas et Yohann.

Enfin, je remercie infiniment Ana pour la patience, les encouragements, le soutien et le reconfort moral qu'elle a su m'accorder tout au long de ces trois années de thèse.

*à Ana et Jean ...*

# Table des matières

**Introduction Générale** **13**

**Nomenclature et méthodologie**

1 Nomenclature . . . . . 21
2 Méthodologie . . . . . 23
2.1 1ere Etape : Choix des facteurs du domaine expérimental . . . . . 23
2.2 2ème Etape : Proposition d'un modèle . . . . . 23
2.3 3ème Etape : Création du plan d'expériences . . . . . 23
2.4 4ème Etape : Estimation des paramètres du modèle . . . . . 23
2.5 5ème Etape : Validation du modèle . . . . . 23

---

**Chapitre 1**
**Apprentissage Statistique**

1.1 Introduction . . . . . 29
1.2 Introduction à l'apprentissage statistique . . . . . 29
1.2.1 Objectifs . . . . . 29
1.2.2 Analyse de la minimisation du risque empirique . . . . . 31
1.3 Régression linéaire . . . . . 34
1.3.1 Moindres carrés . . . . . 34
1.4 Apprentissage de modèles non linéaires par rapport à leurs paramètres - Les Réseaux de Neurones . . . . . 36
1.5 Régularisation . . . . . 37
1.5.1 Early Stopping . . . . . 38
1.5.2 Weight Decay . . . . . 38
1.6 Conclusion . . . . . 40

**Chapitre 2**
**Sensibilité de la construction d'un modèle par rapport aux exemples**

2.1 Introduction . . . . . 45
2.2 Leave-One-Out . . . . . 45
2.3 Leviers . . . . . 46
2.3.1 Interprétation des leviers . . . . . 50
2.3.2 Une répartition parabolique des leviers . . . . . 51
2.3.3 Le calcul des leviers . . . . . 54
2.3.4 Propriétés de la matrice de projection **H** . . . . . 54
2.4 Le Bootstrap . . . . . 55
2.4.1 Le Principe . . . . . 55
2.4.2 Le Bootstrap en régression . . . . . 58
2.4.3 Analyse combinatoire . . . . . 58
2.4.4 Le Leave-Many-Out . . . . . 62
2.5 Relation entre les leviers et la variance bootstrap . . . . . 63
2.5.1 Pour un modèle linéaire . . . . . 65
2.5.2 Pour un modèle affine . . . . . 68
2.5.3 Généralisation en dimension deux . . . . . 70
2.5.4 La variance bootstrap pour des modèles polynomiaux . . . . . 76
2.5.5 Des leviers avec prise en compte de la sortie . . . . . 81
2.6 Conclusion . . . . . 83

**Chapitre 3**
**Choix de modèles**

3.1 Introduction . . . . . 87
3.2 Estimation de l'erreur de généralisation dans le cadre linéaire . . . . . 87
3.3 Estimation de l'erreur de généralisation par des techniques de ré-échantillonnage 93
3.3.1 Validation croisée . . . . . 94
3.3.2 Le Leave-One-Out . . . . . 94
3.3.3 Estimation de l'erreur de généralisation par Bootstrap . . . . . 96
3.4 Choix d'une complexité . . . . . 98
3.4.1 Principe . . . . . 98
3.4.2 Quelques exemples . . . . . 100
3.5 Heuristiques pour l'initialisation des paramètres et pour la régularisation par arrêt prématuré . . . . . 105
3.5.1 Choix du meilleur modèle par leave-one-out virtuel pour une architecture donnée . . . . . 105
3.5.2 Cas particulier des processus déterministes . . . . . 106
3.5.3 Procédure utilisée pour la construction des modèles par bootstrap . . . . 108

3.6 Conclusion . . . . . 110

**Chapitre 4**
**Méthode des plans d'expériences**

4.1 Introduction . . . . . 115
4.2 Estimation des paramètres et prédiction . . . . . 116
4.2.1 Modélisation . . . . . 116
4.2.2 Matrice de variance-covariance de $\boldsymbol{\theta}_{\mathcal{L}}$ . . . . . 117
4.2.3 La prédiction . . . . . 118
4.2.4 La fonction de variance de prédiction . . . . . 119
4.3 Les Plans Optimaux pour modèle linéaire par rapport aux paramètres . . . . . 120
4.3.1 Conception . . . . . 120
4.3.2 Le critère de E-optimalité . . . . . 121
4.3.3 Le critère de A-optimalité . . . . . 121
4.3.4 Le critère de D-optimalité . . . . . 122
4.3.5 Le critère de G-optimalité . . . . . 123
4.3.6 Le critère de J-optimalité . . . . . 123
4.4 Construction des plans optimaux . . . . . 125
4.4.1 L'algorithme d'échange double de Fedorov . . . . . 125
4.4.2 Exemple d'utilisation d'un plan D-optimal . . . . . 126
4.4.3 Les leviers et la D-optimalité . . . . . 130
4.5 Les plans optimaux pour modèles non linéaire par rapport aux paramètres . . . . 130
4.5.1 La D-optimalité locale . . . . . 130
4.5.2 La X-optimalité . . . . . 131
4.6 Conclusion . . . . . 132

**Chapitre 5**
**Apprentissage actif**

5.1 Introduction . . . . . 137
5.2 Principe général . . . . . 137
5.3 La méthode Learner Disagreement by experiment Resampling . . . . . 139
5.3.1 Le Bagging . . . . . 139
5.3.2 Choix de l'exemple à replanifier . . . . . 140
5.3.3 Un exemple didactique simple . . . . . 142
5.4 Comparaison entre méthodes pour la modélisation du processus de Friedman . . 145
5.4.1 La fonction de Friedman . . . . . 145
5.4.2 Procédure de test et résultats . . . . . 147

5.5 Comparaison entre méthodes pour le processus de Homma et Saltelli . . . . . 149
5.5.1 La fonction de Homma et Saltelli . . . . . 149
5.5.2 Choix de l'architecture optimale pour le modèle de Homma et Saltelli . . 151
5.5.3 Procédure de test et résultats . . . . . 151
5.5.4 Zones de replanification . . . . . 154
5.5.5 Temps de calcul . . . . . 156
5.5.6 Évolution de l'erreur de généralisation en fonction du nombre d'exemples 156
5.6 Conclusion . . . . . 159

**Conclusion Générale**

---

**Annexes** **167**

**Annexe A**
**Calcul du gradient de la fonction de coût** **167**

A.1 Calcul du gradient . . . . . 169

**Annexe B**
**Algorithme d'optimisation de Levenberg-Marquardt** **173**

B.1 L'algorithme d'apprentissage de Levenberg-Marquardt . . . . . 175

**Annexe C**
**Article soumis à IEEE Transactions on Neural Networks** **179**

C.1 Introduction . . . . . 181
C.2 Background : D-optimality in experimental design . . . . . 182
C.2.1 Training models from data . . . . . 182
C.2.2 The linear framework . . . . . 183
C.2.3 The non-linear framework . . . . . 184
C.2.4 Algorithmic Construction of D-optimal experimental designs . . . . . 185
C.3 The Active Learning background . . . . . 185
C.4 Design of experiments by LDR-Bagging . . . . . 186
C.4.1 Bagging . . . . . 186
C.4.2 Active Learning by LDR-Bagging . . . . . 187
C.4.3 A simple didactic example . . . . . 187
C.5 Active Learning by LDR-Leave One Out . . . . . 189
C.6 Results . . . . . 190
C.6.1 The Homma & Saltelli benchmark [Saltelli & Homma 92] . . . . . 191

C.6.2 The Friedman benchmark . . . . . 193
C.6.3 The CEA application . . . . . 194
C.7 Conclusion . . . . . 196

**Bibliographie** **199**

**Index** **201**

**Bibliographie** **207**

# Introduction Générale

La simulation numérique est devenue un outil incontournable en recherche et développement pour évaluer par exemple des modèles d'expansion de l'univers, ou pour approcher des solutions optimales d'un stockage de déchets radioactifs ou d'une architecture d'un barrage. Les études de simulation mettent en jeu des outils appelés codes de calcul. Les codes sont constitués de modèles numériques reliant des variables, descriptives de l'état du système simulé, à un certain nombre de paramètres caractérisant, par exemple, les conditions initiales du système ou bien certaines lois ou relations utilisées dans la modélisation. Le simulateur prend ces paramètres en entrée et délivre en sortie les réponses sur les variables d'état du système. Une simulation consiste à fixer un ou plusieurs jeux de variables d'entrée, à réaliser les calculs, puis à analyser les réponses fournies par le simulateur. Malgré les progrès toujours croissants de l'informatique, ces codes de calcul sont complexes et souvent très longs à exécuter. L'utilisateur souhaite, dans ce cas, obtenir un modèle plus simple, plus rapide et qui rende compte le mieux possible des réponses du simulateur. C'est le cas, par exemple, dans les études d'analyses d'incertitudes et de conception robuste où le code de calcul doit être exécuté un très grand nombre de fois.

Parmi les différentes approches de construction de modèles simplifiés, nous n'évoquerons dans ce mémoire que celles fondées sur les méthodes de construction de méta-modèles, modèles de modèles numériques, fondées sur l'apprentissage statistique en relation avec les plans d'expériences pour surfaces de réponse. La construction du modèle simplifié est fondée sur la réalisation d'un certain nombre de simulations appelées expériences numériques. Les expériences simulées étant coûteuses en temps calcul, il est important de disposer de méthodes permettant d'optimiser leur planification. Pour une famille de fonctions retenues pour construire le modèle simplifié, la qualité de représentation du modèle simplifié dépendra principalement du choix de ces expériences.

Des méthodes de planification d'expériences ont été étudiées par des auteurs comme Fisher [Fisher 25] [Fisher 35], Kiefer [Kiefer 59], Box [Box & Draper 87] ou Fedorov [Fedorov 72]. Leurs travaux ont donné lieu à une nouvelle branche des statistiques connue sous le nom de méthodologie des plans d'expériences. La plupart des critères de ces plans utilisent le bruit de mesure qui existe dans le cadre d'expériences réelles. Les expériences simulées ne rentrent pas dans ce contexte de planification d'expériences car les simulations que l'on considère pour créer le modèle simplifié sont déterministes. Si l'on effectue plusieurs simulations avec des jeux de variables d'entrée identiques, on obtient la même réponse. On ne peut donc pas utiliser directement la méthodologie des plans d'expériences. On peut toutefois l'adapter en considérant que ce bruit expérimental tend vers zéro.

D'autres méthodes de planification d'expériences ont été étudiées dans le domaine de l'apprentissage statistique sous le nom d'apprentissage actif [Cohn 94] [MacKay 92a]. Cette approche est fondée sur la notion de comité d'experts (Query by Committee) introduite par [Seung *et al.* 92]. L'apprentissage actif peut être qualifiée de méthode *a posteriori* et aucune hypothèse de bruit expérimental sur les données n'est nécessaire. La plupart des méthodes en apprentissage actif ont été développées en classification supervisée. Dans ce mémoire nous développerons l'apprentissage actif dans le cadre de la régression.

Les méthodes que l'on propose s'appuient sur la théorie de l'apprentissage statistique. Les modèles simplifiés sont construits en utilisant la famille des polynômes, pour les modèles linéaires en leurs paramètres, et la famille des réseaux de neurones, pour les modèles non linéaires en leurs paramètres. Ces derniers permettent de modéliser plus fidèlement la réponse du simu-

lateur grâce à leurs propriétés d'approximation universelle et de parcimonie. Nous proposons une procédure itérative de planification d'expériences numériques LDR (acronyme de Learner Disagreement from experiment Resampling), fondée sur l'apprentissage actif. On procède à la construction de plusieurs modèles simplifiés construits à partir de répliques bootstrap (méthode de ré-échantillonnage) de la base d'exemples. Par une approche de type Bagging, acronyme de Bootstrap Aggregating, on analyse la variance des prédictions des modèles simplifiés et on planifie les nouvelles simulations de manière à réduire cette variance des prédictions. Cette procédure est comparée aux approches classiques des plans d'expériences optimaux et en particulier avec la D-optimalité.

Le chapitre 1 de ce manuscrit présente la théorie de l'apprentissage statistique que nous utiliserons pour créer les modèles dont nous aurons besoin dans l'ensemble de ce mémoire. Cette théorie, introduite par Vapnik [Vapnik & Chervonenkis 71], permet de traiter une grande variété de problèmes, mais nous n'aborderons que l'aspect de la régression ou modélisation statique. Après avoir décrit le principe de l'apprentissage statistique, nous consacrerons la deuxième partie de ce chapitre à son utilisation pour des modèles linéaires en les paramètres, comme les polynômes, puis, dans une troisième partie, aux modèles non linéaires par rapport aux paramètres comme les réseaux de neurones.

Le chapitre 2 est consacré à l'étude de la sensibilité de la construction d'un modèle par rapport aux exemples. Nous définirons la notion de levier, introduite en régression linéaire, qui est une mesure de l'influence d'un exemple, par son retrait de la base d'apprentissage, dans l'estimation des paramètres du modèle. Ceci rappelle la méthode du leave-one-out utilisée en apprentissage statistique pour estimer l'erreur de généralisation du modèle construit. Nous présenterons ensuite une autre méthode, pouvant être utilisée pour des modèles non linéaires par rapport aux paramètres, fondée sur le ré-échantillonnage Bootstrap que nous décrirons. Nous terminerons ce chapitre avec une étude sur les relations entre les leviers et la mesure de l'influence d'un exemple obtenue par bootstrap. Cette étude se limitera à des modèles linéaires par rapport aux paramètres pour nous affranchir des problèmes de minima locaux des fonctions de coût des modèles non linéaires par rapport aux paramètres.

Le chapitre 3 aborde la problématique de choix de modèles. Cette problématique est directement liée à celle de l'estimation de l'erreur de généralisation du modèle construit, qui est l'un des problèmes les plus importants en apprentissage statistique. Nous présenterons, tout d'abord, une méthode d'estimation de l'erreur de généralisation pour des modèles linéaires par rapport aux paramètres fondée sur une analyse spectrale de la matrice d'information. Cette méthode établit un lien intéressant entre l'apprentissage statistique et la méthode des plans d'expériences A-optimaux pour modèles linéaires par rapport aux paramètres. Pour les modèles non linéaires par rapport aux paramètres, comme les réseaux de neurones, nous présenterons d'autres méthodes d'estimation de l'erreur de généralisation fondées sur des techniques de ré-échantillonnages telles que le leave-one-out, la validation croisée et le bootstrap. La problématique de choix de modèles, au sens de la dimension de Vapnik Chervonenkis (choix du degré du polynôme pour un modèle polynomial ou choix du nombre de neurones cachés pour un réseaux de neurones) sera illustrée à travers des exemples simples pour des modèles polynomiaux. Nous terminerons ce chapitre avec quelques considérations concernant l'initialisation des paramètres du modèle et la régularisation par arrêt prématuré de l'apprentissage pour les modèles non linéaires par rapport aux paramètres de type réseaux de neurones.

Le chapitre 4 décrit la théorie des plans d'expériences optimaux. Toute création d'un modèle, par régression linéaire ou non linéaire, nécessite une base d'apprentissage afin d'estimer les paramètres du modèle à construire. La qualité de ce modèle est directement liée à la qualité de représentation de la base d'exemples. Ce chapitre nous permet de décrire la théorie des plans d'expériences optimaux que nous utiliserons pour mener une étude comparative entre les capacités prédictives des modèles construits à partir de plans optimaux et celles des modèles construits à partir des bases planifiées par la méthode d'apprentissage actif que nous proposons. Après quelques rappels liés à l'estimation des paramètres d'un modèle linéaire, nous décrirons les principaux plans d'expériences optimaux pour modèles linéaires par rapport aux paramètres ainsi que la manière de les mettre en oeuvre. Nous montrerons, en fin de chapitre, comment les adapter à des modèles non linéaires par rapport aux paramètres.

Le chapitre 5 présente la méthode d'apprentissage actif LDR (Learner Disagreement from experiment Resampling) que nous proposons dans le cadre de cette thèse. La méthode LDR utilise le ré-échantillonnage des exemples de la base d'apprentissage pour spécifier les nouvelles expériences numériques à effectuer. Nous utiliserons surtout la technique de ré-échantillonnage du Bootstrap et la méthode du Bagging pour proposer une approche itérative de spécification des expériences simulées. Dans une première partie, nous décrirons le principe général de l'apprentissage actif ainsi que du Bagging. Nous illustrerons ensuite la méthode LDR sur des exemples "jouets" simples. Enfin, nous testerons et comparerons, à l'aide des processus de *Friedman* et de *Homma et Saltelli*, les bases d'apprentissages obtenues à partir de la méthode LDR aux plans d'expériences obtenus à partir de la D-optimalité locale.

# Nomenclature et méthodologie

Nous allons définir, dans cette section, la plupart des symboles que nous utiliserons dans l'ensemble de ce mémoire ainsi que la méthodologie générale de construction d'un modèle à partir de données.

# 1 Nomenclature

| Symbole | Description |
|---|---|
| $\mathbf{x}$ | un vecteur des variables du modèle. Ce vecteur contient $d$ élements correspondant aux valeurs des facteurs. Ce vecteur est également appelé entrée $\mathbf{x}$ ou exemple $\mathbf{x}$ |
| $\mathbf{x}_i$ | désigne le vecteur des variables du modèle correspondant à l'exemple $i$. |
| $x_j$ | désigne la valeur du facteur $j$ pour l'exemple courant $\mathbf{x}$. |
| $y(\mathbf{x})$ | désigne la réponse mesurée ou simulée du processus inconnu correspondant à l'entrée $\mathbf{x}$. |
| $U$ | désigne le domaine d'existence des variables. |
| $y_i$ | désigne la valeur mesurée de la grandeur à modéliser pour l'exemple $i$ ($y_i = y(\mathbf{x}_i)$). |
| $N$ | désigne le nombre d'exemples d'un plan d'expériences ou d'une base d'apprentissage. |
| $p$ | désigne le nombre de paramètres du modèle que l'on construit par régression. |
| $\mathcal{L}$ | désigne la base d'apprentissage constituée de $N$ paires d'entrées/sorties $\mathcal{L} = \{(\mathbf{x}_i, y_i), 1 \leq i \leq N\}$ |
| $\mathcal{L}^{*b}$ | désigne la réplique bootstrap $b$ de la base d'apprentissage $\mathcal{L}$. |
| $L_N$ | désigne la matrice d'un plan d'expériences contenant $N$ expériences. l'element $ij$ correspond à la valeur du facteur $j$ pour l'exemple $i$. |
| $\boldsymbol{\phi}(\mathbf{x})$ | désigne le vecteur des variables secondaires. |
| $\phi_i(\mathbf{x})$ | sont les fonctions de régression qui permettent de passer des variables primaires aux variables secondaires. |
| $\mathbf{X}$ | désigne la matrice du modèle ou matrice des effets. L'élement $ij$ de cette matrice correspond à la valeur de la fonction de régression $j$ pour l'exemple $i$. |
| $\mathbf{X'X}$ | désigne la matrice d'information du modèle décrit dans la matrice $\mathbf{X}$. La matrice d'information de Fisher $\sigma^2\mathbf{X'X}$, s'écrit également $\mathbf{X'X}$ dans le cas linéaire avec des erreurs gaussiennes de variance unité. |

| Symbole | Description |
|---|---|
| $\mathbf{X}'\mathbf{X}^{-1}$ | désigne la matrice de dispersion. |
| $\mathbf{X}_{(i)}$ | désigne la matrice du modèle à laquelle on a extrait $\boldsymbol{\phi}(\mathbf{x}_i)$ |
| $\boldsymbol{\theta}$ | désigne le vecteur des paramètres du modèle construit par régression. |
| $\boldsymbol{\theta}_{\mathcal{L}}$ | désigne le vecteur des paramètres qui minimise la fonction de coût d'apprentissage obtenu à partir de la base d'apprentissage complète $\mathcal{L}$. |
| $\Theta$ | désigne l'espace des paramètres. |
| $\theta_{\mathcal{L},j}$ | désigne la valeur de la composante $j$ du vecteur de paramètres $\boldsymbol{\theta}_{\mathcal{L}}$ |
| $\boldsymbol{\theta}_{\mathcal{L}}^{(i)}$ | désigne le vecteur des paramètres du modèle qui minimise la fonction de coût d'apprentissage obtenu à partir de la base $\mathcal{L}$ à laquelle on a extrait l'exemple $i$. |
| $f(\mathbf{x}, \boldsymbol{\theta})$ | désigne la réponse du modèle à l'entrée $\mathbf{x}$ pour le jeu de paramètres $\boldsymbol{\theta}$. |
| $f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}})$ | désigne la réponse du modèle à l'entrée $\mathbf{x}$ après apprentissage sur la base $\mathcal{L}$. |
| $f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}^{*b}})$ | désigne la réponse du modèle à l'entrée $\mathbf{x}$ après apprentissage sur la réplique $b$ de la base $\mathcal{L}$. |
| $f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}}^{(i)})$ | désigne la réponse du modèle à l'entrée $\mathbf{x}$ après apprentissage obtenu à partir de la base $\mathcal{L}$ à laquelle on a extrait l'exemple $i$. |
| $\mathcal{F}$ | désigne la famille de fonctions dont est issu $f$. |
| $J(\boldsymbol{\theta})$ | désigne la fonction de coût d'apprentissage. |
| $\sigma^2$ | désigne la variance du bruit expérimental s'il existe. |
| $s^2(.)$ | désigne l'estimation d'une variance. |
| $s^2(\boldsymbol{\theta}_{\mathcal{L}})$ | désigne l'estimation de la variance de $\boldsymbol{\theta}_{\mathcal{L}}$. |
| $s^2_{loo}(\boldsymbol{\theta}_{\mathcal{L}})$ | désigne l'estimation de la variance de $\boldsymbol{\theta}_{\mathcal{L}}$ par leave-one-out. |
| $s^2_{\mathcal{B}}(\boldsymbol{\theta}_{\mathcal{L}})$ | désigne l'estimation de la variance de $\boldsymbol{\theta}_{\mathcal{L}}$ par bootstrap. |
| $s^2_{\mathcal{VC}}(\boldsymbol{\theta}_{\mathcal{L}})$ | désigne l'estimation de la variance de $\boldsymbol{\theta}_{\mathcal{L}}$ par validation croisée. |

# 2 Méthodologie

Ce paragraphe permet de décrire la méthode utilisée pour la création d'un modèle. Nous considérerons dans la suite du mémoire que la première étape (*choix des facteurs du domaine expérimental*) a été effectuée. De plus, pour la deuxième étape (*Proposition d'un modèle*), nous utiliserons soit la famille de fonctions des polynômes de degré fixé pour les modèles linéaires par rapport aux paramètres, soit la famille de fonctions des réseaux de neurones à nombre de neurones cachés fixés pour les modèles non linéaires par rapport aux paramètres. Les étapes de la démarche méthodologique sont les suivantes :

## 2.1 1ere Etape : Choix des facteurs du domaine expérimental

Une fois que nous avons choisi les facteurs à utiliser, il faut centrer et réduire les différents facteurs choisis afin qu'ils aient tous la même influence dans la construction du modèle. On peut ensuite choisir la forme du domaine d'étude. Il peut être :

- Sphérique : le domaine expérimental est choisi autour d'un point central d'intérêt.
- Cubique : le domaine expérimental est choisi en fonction des plages de variations des différents facteurs.
- Quelconque : un domaine hypersphérique ou hypercubique est utilisé, dans lequel une ou plusieurs parties sont exclues (domaine d'étude possible).

## 2.2 2ème Etape : Proposition d'un modèle

Les modèles choisis sont très variés et dépendent du type de problème étudié : modèles se traduisant par des équations algébriques ou des équations différentielles, modèles qui se traduisent par des équations linéaires ou non linéaires ou bien encore des modèles statiques ou dynamiques etc. Il faut donc choisir un modèle parmi une famille de fonctions. Le problème qui se pose donc est le choix de cette famille de fonctions et le choix de la complexité retenue pour le modèle postulé. Dans tous les cas, ces modèles doivent :

- bien représenter la réponse expérimentale en fonction des variables d'entrées sur le domaine d'étude.
- permettre d'obtenir une estimation satisfaisante (suivant un certain critère) de la grandeur à modéliser.

## 2.3 3ème Etape : Création du plan d'expériences

Cette étape de création du plan d'expériences consiste à déterminer les expériences à réaliser par rapport au choix de modèle décidé à la deuxième étape.

## 2.4 4ème Etape : Estimation des paramètres du modèle

L'estimation des paramètres du modèle diffère suivant la famille de fonctions choisie. Dans le cas des modèles linéaires en leurs paramètres, l'estimation des paramètres met en jeu des techniques d'algèbre linéaire ; dans le cas des modèles non linéaires, des méthodes d'optimisation plus élaborées sont utilisées.

## 2.5 5ème Etape : Validation du modèle

Á la fin de l'étape de validation, il y a deux possibilités :

- soit le modèle est validé, auquel cas il représente bien le phénomène étudié dans le domaine expérimental. L'objectif de la modélisation est atteint : ce modèle peut être mis en oeuvre pour faire de la prévision sur l'ensemble du domaine expérimental.
- soit le modèle n'est pas validé, auquel cas il faut proposer un autre modèle au sein de la même famille de fonctions (changer la complexité du modèle) ou bien proposer une autre famille de fonctions.

# Chapitre 1

# Apprentissage Statistique

**Sommaire**


**1.1 Introduction . . . . . 29**
**1.2 Introduction à l'apprentissage statistique . . . . . 29**
1.2.1 Objectifs . . . . . 29
1.2.2 Analyse de la minimisation du risque empirique . . . . . 31
**1.3 Régression linéaire . . . . . 34**
1.3.1 Moindres carrés . . . . . 34
**1.4 Apprentissage de modèles non linéaires par rapport à leurs paramètres - Les Réseaux de Neurones . . . . . 36**
**1.5 Régularisation . . . . . 37**
1.5.1 Early Stopping . . . . . 38
1.5.2 Weight Decay . . . . . 38
**1.6 Conclusion . . . . . 40**

## 1.1 Introduction

Ce chapitre introduit les techniques d'apprentissage que nous utiliserons dans ce mémoire pour créer nos modèles. La première partie décrit le principe de l'apprentissage statistique. La deuxième partie sera consacrée à son utilisation pour des modèles linéaires en leurs paramètres, et la troisième partie aux modèles non linéaires en leurs paramètres.

## 1.2 Introduction à l'apprentissage statistique

On désigne sous le terme "d'apprentissage statistique" une grande variété de problèmes : classification, régression, modélisation statique ou dynamique, agrégation de données, analyse en composantes principales ou en composantes indépendantes, estimation de probabilités, etc. Dans ce mémoire, nous aborderons le problème de la régression (ou modélisation statique).

Pour modéliser un processus inconnu, nous disposons d'une base d'exemples, ou base d'apprentissage $\mathcal{L}$, constituée de $N$ paires d'entrées / sortie $\mathcal{L} = \{(\mathbf{x}_i, y_i), 1 \leq i \leq N\}$, où $\mathbf{x}_i$ désigne le vecteur des variables du modèle pour l'exemple $i$, et $y_i$ la valeur, mesurée ou simulée, de la grandeur à modéliser. Nous noterons par $y(\mathbf{x})$ la fonction inconnue que l'on cherche à approcher par une expression analytique ($y_i = y(\mathbf{x}_i)$). On supposera dans l'ensemble de ce mémoire que la grandeur à modéliser est scalaire : on cherche donc une application de $\mathcal{R}^d$ dans $\mathcal{R}$, où $d$ est la dimension de l'espace des variables. L'extension de l'ensemble des résultats à des sorties vectorielles est envisageable même s'il reste encore du travail à faire dans ce domaine. On désignera par $U \in \mathcal{R}^d$ le domaine d'existence des variables, appelé également domaine d'étude.

L'objectif de la régression est de trouver une application de $\mathcal{R}^d$ dans $\mathcal{R}$ qui

- rende compte "le mieux possible" (en un sens que nous préciserons plus loin) des données existantes.

- possède une capacité *prédictive*, c'est-à-dire qui soit capable de prédire de manière satisfaisante (selon un critère que nous définirons plus loin) la valeur de la grandeur à modéliser $y$ pour des vecteurs de variables $\mathbf{x}$ qui ne sont pas présents dans la base d'apprentissage.

### 1.2.1 Objectifs

Dans le cadre de la théorie de l'apprentissage statistique [Vapnik & Chervonenkis 71] les valeurs de la grandeur à modéliser $y$ présentes dans la base d'apprentissage sont supposées être des réalisations d'une variable aléatoire de distribution inconnue $P(y|\mathbf{x})$. Cette distribution est liée à la distribution conjointe $P(\mathbf{x}, y)$ par la relation :

$$P(\mathbf{x}, y) = P(\mathbf{x}).P(y|\mathbf{x})$$

On peut donc considérer que la création d'un modèle met en oeuvre trois éléments :

– Un **générateur** ($\mathcal{G}$) de vecteurs $\mathbf{x} \in \mathcal{R}^d$, distribués suivant la loi $P(\mathbf{x})$.

– Un **superviseur** ($\mathcal{S}$) qui donne, pour tout vecteur d'entrée $\mathbf{x}$, la valeur $y$ suivant la distribution conditionnelle $P(y|\mathbf{x})$ inconnue.

– Un **apprenti** ($\mathcal{LM}$ pour "Learning Machine") capable de réaliser une classe de fonctions paramétrées $f(\mathbf{x}, \boldsymbol{\theta}) \in \mathcal{F}$ où $\boldsymbol{\theta}$ appartient à l'espace des paramètres $\Theta$.

Durant l'apprentissage, le générateur $\mathcal{G}$ produit des valeurs de $\mathbf{x}$ suivant la distribution de probabilité $P(\mathbf{x})$. Le superviseur reçoit cette même valeur de $\mathbf{x}$ et répond par la sortie $y$ associée à $\mathbf{x}$ par la distribution de probabilité conditionnelle $P(y|\mathbf{x})$. L'objectif de l'apprentissage est de trouver le vecteur de paramètres $\boldsymbol{\theta}$ tel que la réponse $f(\mathbf{x}, \boldsymbol{\theta})$ donnée par l'apprenti soit "la plus proche possible" de $y$ (figure 1.1).

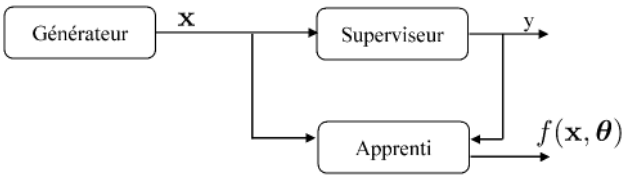



FIG. 1.1 – Schéma de l'apprentissage supervisé

Pour juger de la qualité de l'apprenti, on introduit une "fonction de perte" adaptée à la tâche à apprendre. À titre d'exemples, on utilise fréquemment :

– Pour la classification, où $y$ est une variable binaire qui vaut +1 ou -1 :
$l(f(\mathbf{x}, \boldsymbol{\theta}), y) = I_{f(\mathbf{x},\boldsymbol{\theta})\neq y}$, où $I$ est la fonction indicatrice telle que $I_A = 1$ si $A$ est vrai et $-1$ sinon.

– Pour la régression, où $y$ est un réel :

$$l(f(\mathbf{x}, \boldsymbol{\theta}), y(\mathbf{x})) = (f(\mathbf{x}, \boldsymbol{\theta}) - y(\mathbf{x}))^2 \tag{1.1}$$

Le but de l'apprenti est de prédire la sortie $y$ pour un $\mathbf{x}$ quelconque, sachant que la grandeur à modéliser $y$ obéit à la probabilité conditionnelle $P(y|\mathbf{x})$. L'objectif de l'apprentissage statistique est donc de trouver, parmi une famille de fonctions $\mathcal{F}$, la fonction $f$ qui minimise l'espérance mathématique de la perte, appelée "risque théorique" ou "risque fonctionnel". Pour la régression, on a :

$$R(f) = \int_U (f(\mathbf{x}, \boldsymbol{\theta}) - y(\mathbf{x}))^2 P(\mathbf{x}) d\mathbf{x} \tag{1.2}$$

Bien entendu, ce risque n'est pas calculable, même numériquement.

### 1.2.2 Analyse de la minimisation du risque empirique

Comme nous venons de le décrire, l'objectif de l'apprentissage est de chercher la fonction $f$ appartenant à la famille de fonction $\mathcal{F}$ qui minimise le risque fonctionnel (équation (1.2)) appelé "erreur de généralisation". La probabilité conditionnelle $P(y|\mathbf{x})$ étant inconnue, on ne peut qu'estimer ce risque sur l'ensemble des données dont on dispose. Etant donnée la base d'apprentissage $\mathcal{L} = (\mathbf{x}_i, y_i)_{1 \leq i \leq N}$, on minimise le risque empirique :

$$R_{emp}(f) = \frac{1}{N} \sum_{i=1}^{N} l(f(\mathbf{x}_i, \boldsymbol{\theta}), y_i) \tag{1.3}$$

Le risque empirique est une estimation non biaisée du risque fonctionnel :

$$\forall f, \lim_{N \to \infty} R_{emp}(f) = R(f)$$

Cette convergence issue de la loi des grands nombres n'est pas suffisante. Pour que la solution minimisant le risque empirique converge vers la solution minimisant le risque fonctionnel, il faut que le risque empirique converge uniformément vers le risque fonctionnel :

$$\forall \epsilon > 0, \lim_{N \to \infty} P[\mathtt{sup}_{f \in \mathcal{F}} |R(f) - R_{\mathtt{emp}}(f)| > \epsilon] = 0$$

Vapnik et Chervonenkis [Vapnik & Chervonenkis 71] ont montré que la convergence uniforme est une condition nécessaire et suffisante pour que l'apprentissage soit consistant. Un apprentissage est dit consistant si l'erreur du modèle construit, calculée sur des données nouvelles, converge vers l'erreur du modèle construit, calculée sur les données d'apprentissage, lorsque la taille de l'ensemble d'apprentissage augmente. Pour démontrer la consistance d'un processus d'apprentissage, il faut soit faire une étude exhaustive pour chacune des fonctions $f \in \mathcal{F}$, soit contrôler une caractéristique globale de l'espace des fonctions $\mathcal{F}$ [Grandvalet 00]. C'est cette dernière approche qui a été proposée par Vapnik et Chervonenkis. Ils ont introduit une mesure caractérisant la complexité de l'espace de fonctions $\mathcal{F}$, c'est-à-dire sa capacité de représentation ou de modélisation. Cette mesure, appelée dimension VC, permet de déterminer la vitesse de convergence du risque empirique vers le risque fonctionnel. En notant $h$ la dimension VC, le théorème *fondamental* de Vapnik Chervonenkis détermine les bornes, en probabilité, de l'écart entre le risque fonctionnel et le risque empirique. Par exemple, en classification supervisée les bornes sont déterminées par le théorème suivant.

***Soit $\mathcal{F}$ une famille de fonctions séparatrices de dimension VC $h$, pour toute distribution $P$ et pour toute base $\{(\mathbf{x}_i, y_i)\}_{1 \leq i \leq N}$ obtenue à partir de cette distribution***

***P, l'inégalité suivante est vraie avec la probabilité*** $1-\eta$ ***: [Vapnik 98]***

$$\forall f \in \mathcal{F}, R(f) \leq R_{emp}(f) + \sqrt{\frac{h(log\frac{2N}{h}+1) - log(\frac{\eta}{4})}{N}} \tag{1.4}$$

Le théorème montre que la borne de la différence entre le risque fonctionnel et le risque empirique est croissante en fonction de la dimension $h$. Le *risque* d'un risque empirique très différent du risque fonctionnel est d'autant plus important que la dimension VC est grande. Pour réduire l'importance de cette différence, il faut plus de points dans la base d'apprentissage.

Avant de présenter les apports du théorème, nous allons donner deux exemples illustrant la dimension VC, le premier en classification et le second en régression.

Dans les problèmes de classification supervisée à 2 classes d'éléments dans $R^d$, la dimension VC d'un espace de fonctions séparatrices correspond au nombre maximal de points de $R^d$ pouvant être classés sans erreur par une des séparatrices de $\mathcal{F}$. Par exemple pour un problème à deux classes, combien de points peuvent être classés correctement à coup sûr par une droite dans $\mathcal{R}^2$ ? Trois points non alignés peuvent toujours être séparés par une droite. En revanche, il existe des configurations de plus de trois points qui ne peuvent donner lieu à une séparation sans erreur par une droite. Par exemple, il est impossible de séparer sans erreur le problème à deux classes obtenu à partir des points de la fonction logique XOR. Ce problème n'est pas linéairement séparable par des fonctions indicatrices à suppport affine. Donc, dans ce cas, la dimension VC des indicatrices à support affine dans $\mathcal{R}^2$ est 3. Plus généralement, la dimension VC de la famille des fonctions indicatrices à support affines dans $\mathcal{R}^d$ est égale à $d+1$ [Cover 65] [Burges 98] [Vapnik 98].

En régression, l'interprétation de la dimension VC est moins évidente. Elle est souvent supposée liée explicitement au nombre de degrés de liberté des fonctions de $\mathcal{F}$. Il est souvent évoqué le risque lié aux fonctions sur-paramétrées et la logique est alors de réduire le nombre de paramètres, ce qui n'est pas toujours suffisant pour réduire la dimension VC. Illustrons cela en rappelant l'exemple cité dans [Vapnik 98]. L'exemple porte sur l'espace de fonctions définies dans $R$ à valeurs dans $[-1,+1]$ par la fonction de base $sin(\theta x)$ où $\theta \in R$ représente le degré de liberté sur lequel portera la minimisation du risque empirique. On montre que, quelle que soit la base d'apprentissage, il existe une valeur particulière de $\theta$ interpolant exactement les points. Cette solution optimale de $\theta$ annule donc le risque empirique. Pour ce type de problème, la dimension VC de cette classe de fonctions est infinie alors que l'espace de fonctions ne regroupe que des fonctions à un seul degré de liberté. Donc pour ce cas précis, lorsque $\mathcal{F} = \{sin(\theta x), \theta \in R\}$, l'intervalle de confiance n'est plus borné. Nous avons défini en début de section qu'un modèle était consistant si l'erreur sur des données nouvelles du modèle construit, convergait vers l'erreur sur les données d'apprentissage lorsque le nombre d'exemples de l'ensemble d'apprentissage augmente. Ceci revient à dire qu'un modèle est dit consistant si et seulement si la famille $\mathcal{F}$ dont il est issu est de dimension VC $h$ finie. Dans ce cas, la borne $D(N,h,\eta)$ tend vers zero lorsque $N$ tend vers l'infini.

Le théorème précédent montre que la borne de l'écart entre le risque empirique et le risque

fonctionnel est une fonction croissante de la dimension VC. Ce résultat se généralise pour toute classe de fonctions. La borne $D(N, h, \eta)$ définie par :

$$D(N, h, \eta) = \sqrt{\frac{h(log\frac{2N}{h} + 1) - log(\frac{\eta}{4})}{N}}$$

est une fonction croissante de la dimension VC $h$ et décroissante du nombre d'exemples $N$. Le risque empirique est une fonction décroissante de la dimension VC, un compromis devra être fait sur la valeur de la dimension VC la plus adaptée au problème, de façon à réduire le risque empirique tout en contrôlant la borne $D(N, h, \eta)$. Cette approche est appelée minimisation du risque struturel dans la mesure où elle prend en considération les propriétés de l'espace des fonctions. La figure 1.2 illustre la minimisation du risque structurel.

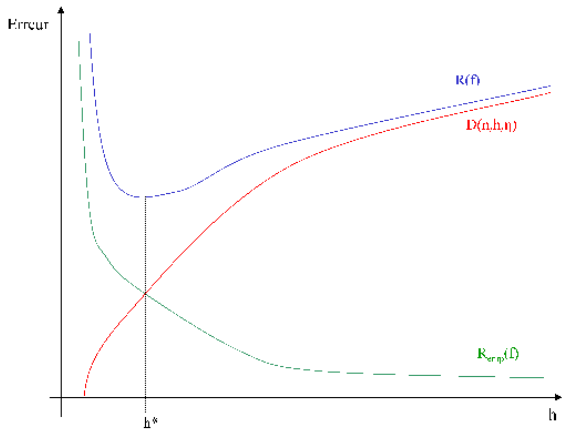



FIG. 1.2 – Compromis entre le risque empirique et le risque fonctionnel

En effet, dans le cadre d'une fonction de perte quadratique, le risque fonctionnel (équation (1.2)) peut se décomposer, pour toute fonction $f$, en deux termes appelés respectivement biais et variance. Le biais correspond à la moyenne sur toutes les bases d'apprentissage possibles du carré de la différence entre la fonction $f(\mathbf{x}, \boldsymbol{\theta})$ (le modèle construit) et $y(\mathbf{x})$ (le processus à modéliser). La variance exprime la sensibilité du modèle à l'ensemble des données utilisées pour l'apprentissage. La Figure 1.2 fait apparaître que pour des valeur de $h$ plus faibles que $h^*$ (solution optimale de la dimension VC) la capacité de modélisation de l'apprenti est insuffisante. On parle alors de sous-paramétrisation du modèle caractérisée par un fort biais. Pour des valeurs de $h$ plus grandes que $h^*$, le risque empirique est faible mais le risque fonctionnel est plus important. Dans cette zone, la sensibilité de l'apprenti par rapport à l'échantillon est importante, l'apprentissage a été fait *par coeur* sans capacité de généralisation. On parle alors de sur-paramétrisation du modèle caractérisée par une forte variance. Le modèle de complexité $h^*$ offre le meilleur compromis.

Mais la dimension VC d'un modèle étant rarement calculable dans la plupart des modèles non linéaires utilisés en classification ou en régression, il est difficile d'obtenir la complexité

optimale. On ne peut donc pas calculer exactement le minimum du risque structurel et on ne peut qu'utiliser des méthodes pour l'estimer. C'est pourquoi les méthodes classiques de régularisation sont encore largement utilisées en apprentissage statistique. Nous définirons certaines d'entre elles à la fin du chapitre et nous verrons leur mise en oeuvre au chapitre 3 qui traite de la sélection de modèles. Une alternative aux techniques *mathématiques* de régularisation repose sur une analyse statistique obtenue par ré-échantillonnage de la base d'apprentissage. Elles sont principalement fondées sur le leave-one-out et le bootstrap. Nous avons utilisé ces techniques ; elles seront donc largement développées dans la suite de ce mémoire.

## 1.3 Régression linéaire

Dans cette partie nous allons traiter le cas où les modèles utilisés par l'apprenti sont linéaires par rapport aux paramètres $\boldsymbol{\theta}$ et rappeler la méthode des moindres carrés.

### 1.3.1 Moindres carrés

Nous considérons l'apprentissage d'une fonction de $\mathcal{R}^d$ dans $\mathcal{R}$ sur un espace de fonctions $\mathcal{F}$ définie par une combinaison linéaire de fonctions $\phi_i(\mathbf{x})$.

$$\mathcal{F} = \{f : \mathcal{R}^d \to \mathcal{R} \ \mid \ f(\mathbf{x}, \boldsymbol{\theta}) = \sum_{i=0}^{p-1} \theta_i \phi_i(\mathbf{x})\}$$

Les $\phi_i(\mathbf{x})$ sont des fonctions non paramétrées, ou à paramètres fixés, d'un ou plusieurs facteurs regroupés au sein du vecteur $\mathbf{x}$, les $\theta_i$ sont les paramètres et $p$ le nombre de paramètres. Nous désignerons par "variables primaires" les variables $\mathbf{x}$ et par "variables secondaires" le vecteur $\boldsymbol{\phi}(\mathbf{x})$, de dimension $p$. Dans ce cas :

$$f(\mathbf{x}, \boldsymbol{\theta}) = \boldsymbol{\phi}(\mathbf{x})' \boldsymbol{\theta}$$

Par exemple, si :

$$f(\mathbf{x}, \boldsymbol{\theta}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1 x_2 + \theta_5 x_3^2$$

nous avons alors, le nombre de facteurs des variables primaires $d = 3$, le nombre de paramètres $p = 6$ et les variables secondaires ou *régresseurs* sont :

$\phi_0(\mathbf{x}) = 1$, $\phi_1(\mathbf{x}) = x_1$, $\phi_2(\mathbf{x}) = x_2$, $\phi_3(\mathbf{x}) = x_1^2$, $\phi_4(\mathbf{x}) = x_1 x_2$ et $\phi_5(\mathbf{x}) = x_3^2$.

Les fonctions $\phi_i(\mathbf{x})$ peuvent être quelconques. On utilise généralement des monômes de degré 0 pour le premier terme ($\forall \mathbf{x}, \phi_0(\mathbf{x}) = 1$) puis de degré croissant en fonction de l'indice $p$. D'autres fonctions de base telles que les fonctions gaussiennes à centre et écart-type fixés, les fonctions splines ou les fonctions ondelettes à translations et dilatations fixées peuvent être également utilisées. Nous utiliserons dans la suite de ce mémoire des modèles linéaires en leurs paramètres

de type polynômes.

Rappelons la fonction perte utilisée en régression :

$$l(f(\mathbf{x}, \boldsymbol{\theta}), y(\mathbf{x})) = (f(\mathbf{x}, \boldsymbol{\theta}) - y(\mathbf{x}))^2$$

A partir d'une base d'exemples $\mathcal{L} = \{(\mathbf{x}_i, y_i), i = 1, \ldots, N\}$, le risque empirique s'exprime donc par

$$R_{emp}(f) = \frac{1}{N} \sum_{i=1}^{N} (f(\mathbf{x}_i, \boldsymbol{\theta}) - y_i)^2$$

Comme la fonction $f(\mathbf{x}, \boldsymbol{\theta})$ est linéaire en $\boldsymbol{\theta}$, le risque empirique s'exprime de façon plus concise par l'expression matricielle suivante en notant $\mathbf{X}$ la matrice (appelée matrice du modèle ou matrice des effets) dont l'élément $X_{ij}$ correspond à la valeur du régresseur $j$ pour l'exemple (expérience) $i$. Cette matrice contient $N$ lignes correspondant au nombre d'exemples de la base d'apprentissage et le nombre $p$ de colonnes est égal au nombre de paramètres du modèle.

$$X_{ij} = \phi_j(\mathbf{x}_i) \rightarrow R_{emp}(f) = \frac{1}{N} (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})'(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

Le minimum existe toujours puisque le risque empirique est une fonction convexe de $\boldsymbol{\theta}$. Par différentiation, on obtient :

$$\frac{\partial R_{emp}(f)}{\partial \boldsymbol{\theta}} = \frac{2}{N} (\mathbf{X}'\mathbf{X}\boldsymbol{\theta} - \mathbf{X}'\mathbf{y}) \tag{1.5}$$

$$\frac{\partial^2 R_{emp}(f)}{\partial \boldsymbol{\theta}^2} = \frac{2}{N} \mathbf{X}'\mathbf{X} \tag{1.6}$$

Si $\mathbf{X}$ est de rang plein (égal au nombre de variables secondaires), la matrice $\mathbf{X}'\mathbf{X}$ est non singulière et la solution optimale, que nous noterons $\theta_{\mathcal{L}}$, est unique :

$$\boldsymbol{\theta}_{\mathcal{L}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$

L'hypothèse d'une matrice d'expériences $\mathbf{X}$ de rang plein ne pose pas de problème particulier en régression, si le nombre d'expériences $N$ est supérieur au nombre de régresseurs. Les régresseurs sont des fonctions différentes et les vecteurs colonnes de $\mathbf{X}$ sont, en général, linéairement indépendants.

Nous reprendrons ce formalisme lors de l'analyse de la sensibilité de l'apprenti par rapport aux exemples (notion de leviers) et lors de la présentation des plans d'expériences optimaux fondées sur l'analyse spectrale de la matrice $\mathbf{X}'\mathbf{X}$, appelée matrice d'information.

## 1.4 Apprentissage de modèles non linéaires par rapport à leurs paramètres - Les Réseaux de Neurones

Contrairement au paragraphe précédent, nous utilisons ici des fonctions non linéaires par rapport à leurs paramètres. De telles fonctions sont de la forme :

$$f(\mathbf{x}, \boldsymbol{\theta}) = \sum_{i=1}^{P} \theta_i \phi_i(\mathbf{x}, \boldsymbol{\Theta_i})$$

où $\boldsymbol{\Theta_i}$ est le vecteur de paramètres ajustables de la fonction $\phi_i$. La fonction $f(\mathbf{x}, \boldsymbol{\theta})$ réalise donc une combinaison linéaire de fonctions paramétrées.

Dans la suite de ce mémoire, nous utiliserons des modèles particuliers, dits "réseaux de neurones à une couche cachée", dans lesquels les fonctions $\phi_i$ sont :

- $\phi_1 = 1$
- $\phi_{i \neq 1} = \frac{1}{1+exp(-\Theta_i' \mathbf{x})}$ (fonction logistique)

Les fonctions $\phi_i$ sont appelées "neurones cachés" ; un tel réseau de neurones est fréquemment représenté graphiquement comme indiqué sur la figure 1.3. Nous utiliserons, dans l'ensemble de ce mémoire, des fonctions $\phi_i$ de type fonction logistique[1].

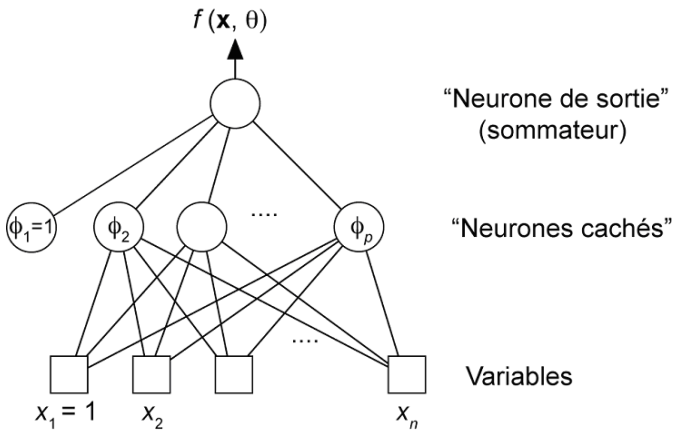



FIG. 1.3 – Représentation graphique d'un réseau de neurones.

[1]La fonction tangente hyperbolique est également fréquemment employée pour les neurones cachés.

Les deux propriétés fondamentales de ces réseaux, qui justifient leurs nombreuses applications, sont les suivantes :

– Toute fonction non linéaire suffisamment régulière peut être approchée uniformément, dans un domaine borné de l'espace des variables, par un réseau de neurones du type représenté sur la figure 1.3 (propriété d'approximation universelle).

– Le nombre de paramètres du réseau varie linéairement avec le nombre de variables, alors qu'il varie exponentiellement avec ce dernier pour les approximateurs linéaires par rapport aux paramètres. La dimension de Vapnik-Chervonenkis des réseaux du type de ceux qui sont représentés sur la figure 1.3 n'est pas calculable exactement ; sa borne inférieure est $O(p^2)$ (où p est le nombre de paramètres), et sa borne supérieure en $O(p^4)$.

La fonction réalisée par un réseau de neurones étant non linéaire par rapport à ses paramètres, le risque empirique n'est pas quadratique par rapport aux paramètres ; il en résulte

– qu'il ne peut pas être trouvé par la méthode des moindres carrés décrite dans le paragraphe précédent. La minimisation du risque empirique nécessite de recourir à des méthodes d'optimisation dont certaines requièrent le calcul du gradient de la fonction de coût par rapport aux paramètres. Ce gradient peut être calculé par un algorithme économe en temps de calcul, dit algorithme de rétropropagation, décrit dans l'annexe A. Une fois le gradient calculé, il est mis en oeuvre au sein d'une méthode d'optimisation. Dans notre travail, nous avons utilisé la méthode de Levenberg-Marquardt, décrite dans l'annexe B,

– que le minimum du risque empirique, déterminé par les méthodes numériques de descente de gradient peut ne pas correspondre au minimum global.

## 1.5 Régularisation

Lorsque nous avons introduit le dilemme biais-variance, nous avons bien souligné que le but de la modélisation était d'obtenir un modèle suffisamment complexe pour apprendre les données sans pour autant s'adapter au bruit de mesure s'il existe, on parle alors de sur-ajustement. La figure 1.2 nous montrait l'évolution de l'erreur de généralisation en fonction de la dimension VC de la famille de fonction considérée. On observe également ce type de profil lorsque l'on fait varier le nombre de cycles d'apprentissage (voir figure 1.4). Il faut donc être en mesure d'interrompre l'apprentissage au bon moment pour éviter le sur-ajustement ou utiliser des fonctions de coût qui limitent le sur-ajustement du modèle.

On peut donc distinguer deux types de méthodes :

– Les méthodes passives : Il s'agit de méthodes de choix de modèles *a posteriori*. On effectue un grand nombre d'apprentissages et parmi les modèles générés de complexités différentes, on retient le modèle le moins sur-ajusté par validation croisée ou par des techniques statistiques.

– Les méthodes actives : On conduit l'apprentissage en évitant de créer des modèles sur-

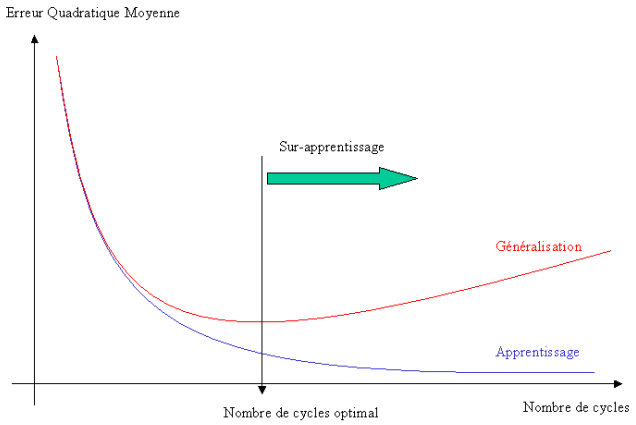



FIG. 1.4 – Evolution de l'erreur de généralisation en fonction du nombre de cycles d'apprentissage.

ajustés. On n'agit pas, cette fois-ci, sur la complexité du modèle mais plutôt sur la limitation de l'amplitude de ses paramètres. On parle alors de méthodes de régularisation [TIKHONOV & ARSENIN 77], [POGGIO *et al.* 85]. Parmi ces méthodes, on peut citer le *Early Stopping* ou arrêt prématuré et le *Weight Decay*.

### 1.5.1 Early Stopping

Le Early Stopping est une méthode de régularisation de l'apprentissage qui a pour but de limiter le sur-ajustement du modèle aux données en faisant un arrêt prématuré de l'apprentissage. Comme pour la figure 1.2 qui décrit l'évolution de l'erreur de généralisation en fonction de la complexité du modèle postulé, on peut trouver le même profil pour l'erreur de généralisation lorsque l'on a en abscisse le nombre de cycles d'apprentissage. Il existe donc également pour le processus d'apprentissage un nombre de cycle optimal qui minimise l'erreur de généralisation. Le but du Early Stopping est de stopper l'apprentissage au voisinage de ce nombre de cycles optimum.

Toute la difficulté réside dans la détermination du moment où l'on doit arrêter l'apprentissage. En pratique, on suit l'évolution de la fonction de coût sur une base de validation constituée d'exemples différents de ceux de la base d'apprentissage. Ceci nous permet d'avoir une estimation de l'évolution de l'erreur de généralisation. On force l'arrêt de l'apprentissage lorsque le coût calculé sur la base de validation commence à croître. Nous exposerons dans le chapitre suivant quelques méthodes pour estimer cette remontée de l'erreur de généralisation.

### 1.5.2 Weight Decay

Une autre méthode consiste à ajouter un terme de pénalisation $\Omega$ à la fonction de coût $J(\boldsymbol{\theta})$, afin de favoriser les solutions les plus régulières :

$$J' = J + \alpha\Omega$$

La fonction de coût $J$ peut être, par exemple, la fonction de coût usuelle des moindres carrés. L'idée est de favoriser les modèles dont la complexité est la plus faible. Le terme $\alpha$, qui est un hyper-paramètre, réalise ce compromis entre le fonction de coût et le terme de pénalisation. Plus $\alpha$ est élevé, plus les modèles de faible complexité sont favorisés ; à l'inverse, plus $\alpha$ est petit et moins le terme de régularisation a d'effet sur l'apprentissage, il y a donc, dans ce cas, un risque de sur-ajustement.

Le terme de pénalisation couramment utilisé est le *Weight Decay* :

$$\Omega = \sum_i \theta_i^2$$

$\theta_i$ correspond aux paramètres du réseau. L'utilisation de ce terme favorise les modèles dont les poids synaptiques sont petits. L'obtention de la valeur de cet hyper-paramètre $\alpha$ relève plus de l'empirisme que de la théorie. Il faut citer néanmoins les travaux de [MacKay 92a] qui propose une démarche statistique solide mais qui conduit à des calculs lourds. En pratique, une démarche heuristique suffit à obtenir de bons résultats. On effectue un grand nombre d'apprentissages avec différentes valeurs de $\alpha$ et on teste les modèles générés sur une base de validation avant d'en choisir le meilleur.

Néanmoins, il est intéressant de noter que les paramètres d'un réseau de neurones n'ont pas tous le même rôle. En effet, la fonction mathématique que réalise un réseau de neurones est la suivante :

$$y = \theta_1 + \sum_{j=2}^{P} \theta_j \phi(\Theta_{1j} + \sum_{i=2}^{n} \Theta_{ij} x_i)$$

où $n$ est le nombre d'entrées du réseau, $P$ le nombre de neurones en couche cachée et $\phi$ la fonction d'activation. On peut donc distinguer 4 types de paramètres :

- les paramètres $\Theta_{1j}$, entre le biais en entrée et les neurones cachés, qui effectuent une translation horizontale des fonctions sigmoïdes.

- les paramètres $\Theta_{ij}$, entre les neurones d'entrées et les neurones cachés, qui changent la fréquence spatiale des fonctions sigmoïdes.

- le paramètre $\theta_1$, entre le biais et le neurone de sortie qui effectue une translation verticale de la réponse du réseau.

– les paramètres $\theta_j$ entre les neurones de la couche cachée et le neurone de sortie qui changent l'amplitude des fonctions sigmoïdes.

Il est donc naturel d'introduire des hyper-paramètres différents pour ces différents paramètres. Il est donc préférable d'utiliser la fonction de coût suivante [MacKay 92a] :

$$J' = J + \alpha_0 \sum_{\Pi_0} \theta_i^2 + \alpha_1 \sum_{\Pi_1} \theta_i^2 + \alpha_2 \sum_{\Pi_2} \theta_i^2$$

où $\Pi_0$ représente l'ensemble des paramètres entre le biais et les neurones cachés, où $\Pi_1$ représente l'ensemble des paramètres entre les neurones d'entrée et les neurones cachés et $\Pi_2$ l'ensemble des paramètres entre les neurones cachés et le neurone de sortie. Dans ce cas là, il est préférable également de déterminer la valeur des hyper-paramètres en effectuant un grand nombre d'apprentissages en changeant la valeur de ces paramètres et de choisir le meilleur modèle en utilisant une base de validation.

## 1.6 Conclusion

Ce chapitre nous a permis d'introduire la théorie de l'apprentissage statistique que nous utiliserons dans l'ensemble de ce mémoire pour créer les modèles dont nous aurons besoin pour notre étude. Nous avons distingué deux familles de modèles : les modèles linéaires par rapport aux paramètres comme les polynômes, et les modèles non linéaires par rapport aux paramètres comme les réseaux de neurones.

Lorsque l'on utilise des modèles linéaires, la fonction de coût des moindres carrés a un minimum unique, qui est facilement obtenu par des méthodes d'algèbre linéaire. La fonction de coût des moindres carrés n'est pas convexe en ce qui concerne les modèles non linéaires en leurs paramètres. La minimisation de la fonction de coût doit être controlée afin d'éviter les minima locaux. La recherche d'un minimum nécessite la mise en oeuvre de méthodes utilisant le gradient de la fonction de coût. Dans certains cas, les méthodes du second ordre de type quasi-Newton, qui mettent en oeuvre la hessienne, sont nécessaires et il faut parfois avoir recours à des algorithmes stochastiques pour approcher un minimum global parmi une multitude de minima locaux.

# Chapitre 2

# Sensibilité de la construction d'un modèle par rapport aux exemples

**Sommaire**


**2.1 Introduction . . . . . 45**
**2.2 Leave-One-Out . . . . . 45**
**2.3 Leviers . . . . . 46**
2.3.1 Interprétation des leviers . . . . . 50
2.3.2 Une répartition parabolique des leviers . . . . . 51
2.3.3 Le calcul des leviers . . . . . 54
2.3.4 Propriétés de la matrice de projection **H** . . . . . 54
**2.4 Le Bootstrap . . . . . 55**
2.4.1 Le Principe . . . . . 55
2.4.2 Le Bootstrap en régression . . . . . 58
2.4.3 Analyse combinatoire . . . . . 58
2.4.4 Le Leave-Many-Out . . . . . 62
**2.5 Relation entre les leviers et la variance bootstrap . . . . . 63**
2.5.1 Pour un modèle linéaire . . . . . 65
2.5.2 Pour un modèle affine . . . . . 68
2.5.3 Généralisation en dimension deux . . . . . 70
2.5.4 La variance bootstrap pour des modèles polynomiaux . . . . . 76
2.5.5 Des leviers avec prise en compte de la sortie . . . . . 81
**2.6 Conclusion . . . . . 83**

## 2.1 Introduction

Dans ce chapitre, nous abordons l'analyse de sensibilité du modèle par rapport aux exemples de la base d'apprentissage. Lorsque l'on a conçu un modèle à partir d'une base d'apprentissage, on peut estimer l'influence d'un exemple $i$ sur les paramètres du modèle en répondant à la question suivante : **que serait-il advenu au modèle si cet exemple n'avait pas fait partie de la base d'apprentissage ?**

Pour répondre à la question précédente, nous rappellerons la notion de **levier**, introduite en régression linéaire, fondée sur le calcul de l'effet sur les paramètres du modèle, du retrait d'un exemple de la base d'apprentissage. Le principe de cette méthode est équivalent à celui du **Leave-One-Out** utilisé en apprentissage statistique pour estimer l'erreur de généralisation. Après ces rappels, nous présenterons une méthode différente de celle des leviers, pouvant être utilisée pour des modèles non linéaires par rapport aux paramètres, fondée sur le **Bootstrap**, technique statistique de ré-échantillonnage qui sera par la suite largement utilisée et étendue à la planifications active d'expériences simulées.

Dans l'ensemble de ce chapitre, nous utiliserons des modèles linéaires en leurs paramètres. Cela nous permettra de nous affranchir des problèmes de minima locaux des fonctions de coût de modèles non linéaires et d'établir des résultats indépendants de la problématique de choix de modèles. Nous pourrons ainsi valider les différentes méthodes dans le cadre linéaire avant de les adapter au cadre non linéaire.

Nous introduirons tout d'abord le leave-one-out et les leviers ; nous rappellerons ensuite le bootstrap, et décrirons l'utilisation que nous en ferons dans la suite du mémoire. Ces notions seront ensuite utilisées au chapitre suivant, qui traite du choix de modèles. Nous terminerons, enfin, avec une étude sur les corrélations entre les leviers et la variance bootstrap.

## 2.2 Leave-One-Out

Le Leave-One-Out (LOO) est une méthode utilisée en apprentissage statistique pour estimer l'erreur de généralisation. Le LOO consiste à répéter plusieurs apprentissages, chacun étant réalisé après avoir retiré un exemple de la base d'apprentissage. Pour chaque apprentissage, on calcule l'erreur sur l'exemple qui a été retiré. Il y a donc autant d'apprentissages et d'erreurs calculées que d'exemples dans la base d'apprentissage. L'erreur de généralisation est estimée à partir de la moyenne empirique des erreurs calculées. Par son principe, le LOO est proche de la technique statistique dite du *Jackknife* introduite par Tukey qui permet de réduire le biais d'un estimateur [Saporta 90][2].

En notant $l(f(\mathbf{x_i}, \boldsymbol{\theta}), y_i)$ la fonction de perte liée à l'estimation de l'exemple $i$ et $f(\mathbf{x}, \boldsymbol{\theta}^{(i)})$ le modèle obtenu par apprentissage de la base d'exemples sans l'exemple $i$, l'estimation de l'erreur de généralisation par LOO s'exprime par :

$$R_{loo}(f) = \frac{1}{N} \sum_{i=1}^{N} l(f(\mathbf{x}_i, \boldsymbol{\theta}^{(i)}), y_i) \tag{2.1}$$

[2]En supposant un biais inversement proportionnel au nombre d'exemples.

où $N$ désigne le nombre d'exemples.

Lorsque le coût $l(.,.)$ est égal à l'écart quadratique entre l'estimation $f(\mathbf{x}, \boldsymbol{\theta})$ et la valeur désirée $y$, l'erreur de généralisation par Leave One Out est alors :

$$R_{loo}(f) = \frac{1}{N} \sum_{i=1}^{N} (f(\mathbf{x}_i, \boldsymbol{\theta}^{(i)}) - y_i)^2$$

Le LOO présente des propriétés intéressantes. Dans [Luntz & Brailovsky 69], il est montré que l'estimation de l'erreur de généralisation est non biaisée : l'espérance de $R_{loo}$ calculée sur tous les échantillons possibles de taille $N$ est égale à l'espérance de l'erreur de généralisation calculée pour les échantillons de taille $N-1$. Certains travaux en cours, comme ceux de Eliseff [Bousquet & Elisseeff 02], portent sur l'analyse de l'estimation de l'erreur de généralisation par LOO en introduisant des notions caractérisant la stabilité des algorithmes d'apprentissage.

Nous allons montrer dans les sections suivantes, pour des modèles linéaires en les paramètres (cas des polynômes par exemple), que le LOO ne nécessite qu'un seul apprentissage. Il correspond alors à un **Leave One Out Virtuel**. Dans ce cas, la méthode permet de représenter rigoureusement l'effet du retrait d'un exemple sur l'estimation des paramètres du modèle. Cet effet dépend du **levier**, mesure de l'influence d'un exemple sur la construction du modèle que nous définirons au prochain paragraphe.

## 2.3 Leviers

Les leviers sont définis en régression linéaire, c'est-à-dire pour des modèles linéaires par rapport aux paramètres, conçus par la méthode des moindres carrés (rappelée au chapitre précédent). Considérons donc le cas où le modèle est linéaire par rapport aux paramètres :

$$f(\mathbf{x}, \boldsymbol{\theta}) = \boldsymbol{\phi}(\mathbf{x})' \boldsymbol{\theta}$$

En reprenant les mêmes notations que celles du chapitre précédent, c'est à dire en notant $\mathbf{X}$ la matrice des effets, rappelons la solution $\boldsymbol{\theta}_{\mathcal{L}}$ des moindres carrés obtenue en utilisant la totalité des exemples de la base $\mathcal{L}$ :

$$\boldsymbol{\theta}_{\mathcal{L}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} \tag{2.2}$$

Notons par $\mathbf{X}_{(i)}$ la matrice des effets dont nous avons extrait l'exemple $i$, ce qui correspond à ôter la ligne $\boldsymbol{\phi}(\mathbf{x}_i)'$ de la matrice $\mathbf{X}$. Nous voulons déterminer l'influence de ce retrait sur l'estimation par moindres carrés des paramètres d'un modèle linéaire.

Faisons la même estimation de $\boldsymbol{\theta}_{\mathcal{L}}$ à partir de la matrice $\mathbf{X}_{(i)}$ en rappelant que $N$ désigne le nombre d'exemples.

$$\mathbf{X}'\mathbf{X} = \sum_{k=1}^{N} \boldsymbol{\phi}(\mathbf{x}_k)\boldsymbol{\phi}(\mathbf{x}_k)' \rightarrow \mathbf{X}'_{(i)}\mathbf{X}_{(i)} = \mathbf{X}'\mathbf{X} - \boldsymbol{\phi}(\mathbf{x}_i)\boldsymbol{\phi}(\mathbf{x}_i)' \tag{2.3}$$

Utilisons le lemme d'inversion [Rao & Toutenburg 95], pour exprimer $\mathbf{X}_{(i)}^{'}\mathbf{X}_{(i)}$ en fonction de $\mathbf{X}'\mathbf{X}$ :

**Lemme d'inversion**

soit $\mathbf{A}$ une matrice carré inversible et deux vecteurs $\mathbf{a}$ et $\mathbf{b}$ tel que $1 + \mathbf{b}'\mathbf{A}^{-1}\mathbf{a} \neq 0$ ; alors :

$$(\mathbf{A} + \mathbf{ab}')^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{ab}'\mathbf{A}^{-1}}{1 + \mathbf{b}'\mathbf{A}^{-1}\mathbf{a}} \tag{2.4}$$

En prenant $\mathbf{A} = (\mathbf{X}'\mathbf{X})$, $\mathbf{a} = -\boldsymbol{\phi}(\mathbf{x}_i)$ et $\mathbf{b} = \boldsymbol{\phi}(\mathbf{x}_i)$ nous obtenons :

$$(\mathbf{X}_{(i)}^{'}\mathbf{X}_{(i)})^{-1} = (\mathbf{X}'\mathbf{X})^{-1} + \frac{(\mathbf{X}'\mathbf{X})^{-1}\boldsymbol{\phi}(\mathbf{x}_i)\boldsymbol{\phi}(\mathbf{x}_i)^{'}(\mathbf{X}'\mathbf{X})^{-1}}{1 - \boldsymbol{\phi}(\mathbf{x}_i)^{'}(\mathbf{X}'\mathbf{X})^{-1}\boldsymbol{\phi}(\mathbf{x}_i)} \tag{2.5}$$

Le terme $\boldsymbol{\phi}(\mathbf{x}_i)^{'}(\mathbf{X}'\mathbf{X})^{-1}\boldsymbol{\phi}(\mathbf{x}_i)$ correspond à l'élément $h_{ii}$ de la matrice chapeau $\mathbf{H} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$ [3]. Cet élément diagonal $i$ de la matrice $\mathbf{H}$ est aussi appelé levier de l'exemple $i$. Nous rappellerons ses propriétés à la fin de cette section, au paragraphe 2.3.4, mais nous pouvons déjà préciser que les leviers vérifient l'inégalité suivante : $0 \leq h_{ii} \leq 1$.

On a donc :

$$(\mathbf{X}_{(i)}^{'}\mathbf{X}_{(i)})^{-1} = (\mathbf{X}'\mathbf{X})^{-1} + \frac{(\mathbf{X}'\mathbf{X})^{-1}\boldsymbol{\phi}(\mathbf{x_i})\boldsymbol{\phi}(\mathbf{x_i})^{'}(\mathbf{X}'\mathbf{X})^{-1}}{1 - h_{ii}}$$

En notant $\boldsymbol{\theta}_{\mathcal{L}}^{(i)}$ les paramètres estimés à partir de la matrice d'expériences $\mathbf{X}_{(i)}$ et par $\hat{y}_i = \boldsymbol{\phi}(\mathbf{x_i})^{'}\boldsymbol{\theta}_{\mathcal{L}}$ l'estimation associée à l'exemple $i$, nous obtenons :

[3] $\theta_{\mathcal{L}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$ d'où $\hat{\mathbf{y}} = \mathbf{H}\mathbf{y}$. Cette matrice s'appelle la matrice chapeau car lorsque l'on multiplie à gauche le vecteur des sorties $\mathbf{y}$ par cette matrice, on obtient le vecteur des sorties estimées souvent noté $\hat{\mathbf{y}}$. Ce projecteur est fondamental dans le cadre de la régression puisqu'il établit un lien direct entre la sortie désirée et la sortie estimée.

$$
\begin{aligned}
\boldsymbol{\theta}_{\mathcal{L}}^{(i)} &= (\mathbf{X}_{(i)}^{'}\mathbf{X}_{(i)})^{-1}\mathbf{X}_{(i)}^{'}\mathbf{y}_{(i)} \\
&= [(\mathbf{X}'\mathbf{X})^{-1} + \frac{(\mathbf{X}'\mathbf{X})^{-1}\boldsymbol{\phi}(\mathbf{x_i})\boldsymbol{\phi}(\mathbf{x_i})^{'}(\mathbf{X}'\mathbf{X})^{-1}}{1-h_{ii}}](\mathbf{X}'\mathbf{y} - \boldsymbol{\phi}(\mathbf{x_i})y_i) \\
&= \boldsymbol{\theta}_{\mathcal{L}} + \frac{(\mathbf{X}'\mathbf{X})^{-1}\boldsymbol{\phi}(\mathbf{x_i})\boldsymbol{\phi}(\mathbf{x_i})^{'}\boldsymbol{\theta}_{\mathcal{L}}}{1-h_{ii}} - (\mathbf{X}'\mathbf{X})^{-1}\boldsymbol{\phi}(\mathbf{x_i})y_i - \frac{(\mathbf{X}'\mathbf{X})^{-1}\boldsymbol{\phi}(\mathbf{x_i})h_{ii}y_i}{1-h_{ii}} \\
&= \boldsymbol{\theta}_{\mathcal{L}} + \frac{(\mathbf{X}'\mathbf{X})^{-1}\boldsymbol{\phi}(\mathbf{x_i})\hat{y}_i}{1-h_{ii}} - (\mathbf{X}'\mathbf{X})^{-1}\boldsymbol{\phi}(\mathbf{x_i})y_i(1 + \frac{h_{ii}}{1-h_{ii}}) \\
\boldsymbol{\theta}_{\mathcal{L}}^{(i)} &= \boldsymbol{\theta}_{\mathcal{L}} + \frac{(\mathbf{X}'\mathbf{X})^{-1}\boldsymbol{\phi}(\mathbf{x_i})\hat{y}_i}{1-h_{ii}} - (\mathbf{X}'\mathbf{X})^{-1}\boldsymbol{\phi}(\mathbf{x_i})y_i\frac{1}{1-h_{ii}}
\end{aligned}
$$

En notant $r_i$ le résidu de l'exemple i, $r_i = \hat{y}_i - y_i = \boldsymbol{\phi}(\mathbf{x_i})^{'}\boldsymbol{\theta}_{\mathcal{L}} - y_i$, nous pouvons exprimer l'effet du retrait de l'exemple $i$ sur les paramètres du modèle. Cette formule a été établie dans [Miller 64] :

$$\boldsymbol{\theta}_{\mathcal{L}} - \boldsymbol{\theta}_{\mathcal{L}}^{(i)} = (\mathbf{X}'\mathbf{X})^{-1}r_i\frac{\boldsymbol{\phi}(\mathbf{x_i})}{1-h_{ii}} \tag{2.6}$$

L'exemple $i$ est d'autant plus influent sur l'estimation des paramètres $\boldsymbol{\theta}$ que son résidu est grand ou son levier proche de 1.

L'effet du retrait d'un exemple peut également être exprimé en fonction de l'erreur d'apprentissage sur l'exemple $i$ et de son levier. Soit $r_j^{(i)}$ l'erreur faite par le modèle de paramètres $\boldsymbol{\theta}_{\mathcal{L}}^{(i)}$ sur l'exemple $j$ présent dans l'ensemble d'apprentissage.

$$
\begin{aligned}
r_j^{(i)} &= y_j - \boldsymbol{\phi}(\mathbf{x}_j)^{'}\boldsymbol{\theta}_{\mathcal{L}}^{(i)} \\
&= y_j - \boldsymbol{\phi}(\mathbf{x}_j)^{'}(\boldsymbol{\theta}_{\mathcal{L}} - (\mathbf{X}'\mathbf{X})^{-1}\boldsymbol{\phi}(\mathbf{x_i})\frac{r_i}{1-h_{ii}}) \\
r_j^{(i)} &= r_j + \frac{h_{ij}}{1-h_{ii}}r_i
\end{aligned}
$$

Soit $r_i^{(i)}$ l'erreur faite sur l'exemple $i$ par le modèle construit sans l'exemple $i$. On retrouve cette formule dans [Miller 90]

$$r_i^{(i)} = \frac{r_i}{1-h_{ii}} \tag{2.7}$$

Cette équation traduit l'effet du retrait d'un exemple de la base d'apprentissage sur l'erreur de prédiction faite par le modèle. Comme nous venons de le montrer, cette relation est exacte dans le cadre linéaire. La méthode porte alors le nom de LOO virtuel dans la mesure où il n'a pas

été nécessaire de réaliser $N$ apprentissages pour estimer l'erreur de généralisation. L'ensemble de ces résultats peuvent se retrouver dans [Belsley *et al.* 80], ouvrage très complet sur le sujet.

Nous allons maintenant analyser la variance des paramètres du modèle sur l'échantillon obtenu par LOO. On rappelle que l'échantillon comporte un nombre d'estimation des paramètres égal au nombre d'exemples $N$. Posons :

$$\begin{aligned} \alpha_i &= \frac{r_i}{1-h_{ii}} \ \texttt{pour} \ i = 1, 2, \ldots, N \\ \mathbf{z}_i &= \boldsymbol{\phi}(\mathbf{x}_i)\alpha_i \end{aligned}$$

La matrice $\mathbf{Z}$ est donc égale à la matrice $\mathbf{X}$ modifiée en multipliant les lignes $\mathbf{x}_i$ par $\alpha_i$. La matrice de variance-covariance de $\boldsymbol{\theta}$, dépend de la matrice de variance-covariance des vecteurs lignes $\mathbf{z}_i$ de $\mathbf{Z}$. En désignant par $\mathbf{g}$ le vecteur dont la composante $i$ est la somme des éléments de la colonne $i$ de la matrice $\mathbf{Z}$ divisée par le nombre d'exemples $N$, et par $s^2_{loo}(\mathbf{z})$ la matrice de variance-covariance des vecteurs lignes $\mathbf{z}_i$ de $\mathbf{Z}$ obtenue par leave-one-out, on a :

$$\begin{aligned} \mathbf{g} &= \frac{1}{N}\mathbf{Z}'\mathbf{1}_N \ \texttt{où} \ \mathbf{1}'_N = (1, 1, \ldots, 1) \\ s^2_{loo}(\mathbf{z}) &= \frac{1}{N}\mathbf{Z}'\mathbf{Z} - \mathbf{g}\mathbf{g}' \end{aligned}$$

En notant $\boldsymbol{\alpha}$ le vecteur $(\alpha_1, \alpha_2, \ldots, \alpha_N)$ et $\texttt{diag}(\boldsymbol{\alpha})$ la matrice diagonale de termes $\alpha_i$, on a :

$$\begin{aligned} \mathbf{Z} &= \texttt{diag}(\boldsymbol{\alpha})\mathbf{X} \\ \Rightarrow s^2_{loo}(\mathbf{z}) &= \frac{1}{N}\mathbf{X}'\texttt{diag}^2(\boldsymbol{\alpha})\mathbf{X} - \frac{1}{N^2}\mathbf{X}'\boldsymbol{\alpha}\mathbf{1}\mathbf{1}'\boldsymbol{\alpha}'\mathbf{X} \\ &= \frac{1}{N}\mathbf{X}'[\texttt{diag}^2(\boldsymbol{\alpha}) - \frac{1}{N}\boldsymbol{\alpha}\boldsymbol{\alpha}']\mathbf{X} \end{aligned}$$

En notant $\boldsymbol{\Lambda}$ la matrice $\texttt{diag}^2(\boldsymbol{\alpha})/N - \boldsymbol{\alpha}\boldsymbol{\alpha}'/N^2$, on a :

$$s^2_{loo}(\mathbf{z}) = \mathbf{X}'\boldsymbol{\Lambda}\mathbf{X} \tag{2.8}$$

$$(\Lambda)_{ij} = \alpha_i^2\delta_{ij}/N - \alpha_i\alpha_j/N^2 \tag{2.9}$$

Puis à partir de l'équation 2.6, la matrice de covariance du vecteur $\boldsymbol{\theta}_{\mathcal{L}}$ s'exprime alors par :

$$s^2_{loo}(\boldsymbol{\theta}_{\mathcal{L}}) = (\mathbf{X}'\mathbf{X})^{-1}s^2_{loo}(\mathbf{z})(\mathbf{X}'\mathbf{X})^{-1} \tag{2.10}$$

$$= (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\boldsymbol{\Lambda}\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1} \tag{2.11}$$

En introduisant la pseudo inverse $\mathbf{X}^+$ de $\mathbf{X}$ définie par $\mathbf{X}^+ = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$ la matrice de covariance de $\boldsymbol{\theta}$ estimée par Leave-One-Out est :

$$s^2_{loo}(\boldsymbol{\theta}_{\mathcal{L}}) = \mathbf{X}^+\boldsymbol{\Lambda}(\mathbf{X}^+)' \tag{2.12}$$

Si l'on suppose que les résidus $r_i$ sont aléatoires, indépendants, centrés et de matrice de variance co-variance $\sigma^2\mathbf{I}$, l'estimation de la variance des paramètres du modèle de régression linéaire est :

$$
\begin{aligned}
s^2_{loo}(\boldsymbol{\theta}_{\mathcal{L}}) &= \mathbf{X}^+(\sigma^2\mathbf{I})(\mathbf{X}^+)' && (2.13)\\
&= \sigma^2(\mathbf{X}'\mathbf{X})^{-1} && (2.14)\\
&= s^2(\boldsymbol{\theta}_{\mathcal{L}}) && (2.15)
\end{aligned}
$$

On peut donc retrouver cette expression à partir de l'équation 2.12 en supposant $\Lambda = \sigma^2 I$, ce qui revient à négliger les termes croisés $\alpha_i\alpha_j/N$ représentant la corrélation des composantes $\alpha_i = r_i/(1-h_{ii})$ si on les suppose centrées. Lorsque l'on fait la même hypothèse sur les résidus corrigés $\alpha_i$ que sur les résidus, on retrouve l'expresssion de la matrice de variance-covariance de $\boldsymbol{\theta}_{\mathcal{L}}$.

### 2.3.1 Interprétation des leviers

Les leviers, rappelons-le, sont les termes diagonaux de la matrice chapeau $\mathbf{H} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$. Nous rappelerons les propriétés de la matrice chapeau au paragraphe 2.3.4. La somme des leviers est égale au nombre $p$ de degrés de liberté du modèle : $\sum_{i=1}^{N} h_{ii} = p$. Le levier relatif à l'exemple $i$, peut donc être interprété comme la fraction des degrés de liberté que le modèle utilise pour s'ajuster à l'exemple $i$ [Monari & Dreyfus 00].

D'après l'équation (2.7), nous pouvons distinguer plusieurs cas :

- Si tous les leviers sont égaux, leur valeur sera $\frac{p}{N}$ [4] où $p$ est le nombre de paramètres du modèle et $N$ le nombre d'exemples. Chaque exemple mobilise une fraction $\frac{p}{N}$ des paramètres du modèle et ils ont tous la même influence. Le modèle n'est spécialisé sur aucun exemple en particulier, ce qui réduit le risque de sur-apprentissage.

- Si un levier est nul, alors le modèle ne consacre aucun degré de liberté à l'exemple correspondant : celui-ci n'a aucune influence sur l'erreur faite par le modèle. Comme nous pouvons le voir à l'équation (2.7), si $h_{ii}$ est nul, l'appartenance ou non de l'exemple $i$ à la base d'apprentissage ne modifiera pas l'erreur faite sur l'exemple $i$.

- Si un levier est très proche de 1, alors l'exemple a une très forte influence sur l'estimation des paramètres du modèle si le résidu associé n'est pas nul. Dans ce cas, si l'exemple ne fait pas partie de la base d'apprentissage, alors l'erreur faite par le modèle sur cet exemple est élevée.

Poursuivons notre étude sur l'interprétation des leviers en utilisant un exemple simple. La figure 2.1 représente quelques points d'un processus inconnu.

Il existe deux points singuliers sur ce processus. Le point 1, qui est un point aberrant. On pourrait croire qu'il a une forte influence sur l'estimation des paramètres, or nous allons voir que ce n'est pas le cas et qu'il est en réalité peu influent et a un levier faible. Au contraire, le point 2, qui se trouve être dans la continuité de la droite de régression est un point très influent et a un levier important.

En effet, dans la définition des leviers, on peut remarquer que la matrice de projection $\mathbf{H}$ dépend uniquement de la matrice $\mathbf{X}$ qui correspond au placement des différents points dans le

[4] Nous verrons également dans les propriétés de la matrice H que $tr(H) = rg(X) = p$ (paragraphe 2.3.4)

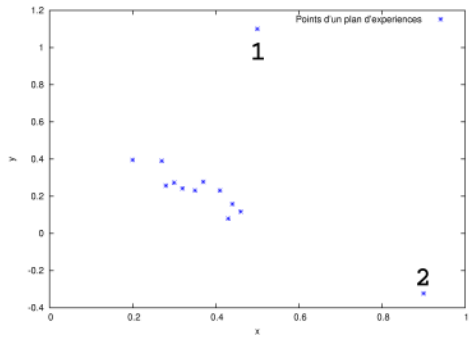



FIG. 2.1 – Mesures effectuées sur un processus à une variable

domaine d'étude. Il faut donc prendre en considération uniquement la projection des points sur l'axe des abscisses. Dans cet espace des entrées, le point 2 est très éloigné des autres points ; il a un levier élevé, comme le montre la figure 2.2. Nous devons souligner que, d'après ces remarques, le point 2 est important uniquement parce qu'il est au bord du domaine. La valeur du levier du point est indépendante de la valeur de $y$ ; il est donc inutile de prendre en considération la position du point par rapport à la droite de régression.

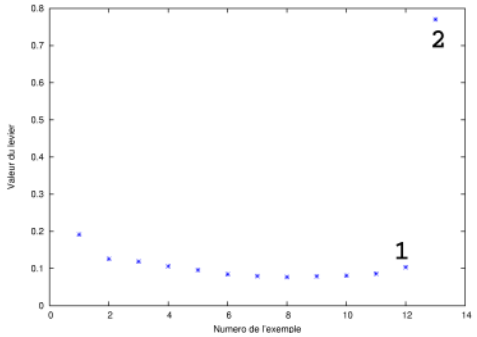



FIG. 2.2 – Leviers des points correspondant au mesures du processus de la figure 2.1

### 2.3.2 Une répartition parabolique des leviers

Dans cette partie nous allons illustrer, sur des cas simples, la répartition des leviers et leur utilisation pour estimer la variance des prédictions d'un modèle linéaire par rapport aux paramètres et en les variables $\mathbf{x}$. Cette étude préliminaire nous permettra par la suite de conduire une analyse comparative entre l'estimation de la variance des prédictions par levier et celle par bootstrap.

Le levier de l'exemple $i$ a pour expression $h_{ii} = \mathbf{x}_i^{'}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{x}_i$. Les points $\mathbf{x} \in R^d$ correspondant à la même valeur de levier, se situent donc sur des ellipsoides car nous avons supposé la matrice $(\mathbf{X}'\mathbf{X})$ inversible. Les leviers du point $\mathbf{x} \in R^d$ se situent donc sur un paraboloide.

Nous allons illustrer le cas $d = 1$, c'est à dire un modèle à une variable $\mathbf{x}$ : $f(x, \boldsymbol{\theta}) = \theta_0 + \theta_1 x$. Les leviers exprimés en fonction de $x$ se situent donc sur une parabole. La base d'apprentissage

est un échantillon de $N$ éléments d'une distribution inconnue. La matrice $\mathbf{X}$ a $N$ lignes et deux colonnes. Les éléments de la première colonne sont des 1 afin d'estimer le terme de biais. Nous voulons calculer les éléments diagonaux de la matrice $\mathbf{H} = \mathbf{X}(\mathbf{X'X})^{-\mathbf{1}}\mathbf{X'}$.

$$\mathbf{X} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & \vdots \\ 1 & x_N \end{pmatrix}$$

d'où

$$\mathbf{X'X} = \begin{pmatrix} N & \sum_{i=1}^{N} x_i \\ \sum_{i=1}^{N} x_i & \sum_{i=1}^{N} x_i^2 \end{pmatrix} = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$$

Le déterminant de cette matrice est :

$$det(\mathbf{X'X}) = N\sum_{i=1}^{N} x_i^2 - (\sum_{i=1}^{N} x_i)^2 \tag{2.16}$$

Les $x_i$ sont distribués suivant une loi inconnue. Soit $s^2$ une estimation de la variance $\sigma^2$ de la distribution des $x_i$, obtenue à partir de l'échantillon $\mathbf{X}$. Nous avons alors :

$$s^2(x) =< x^2 > - < x >^2$$

où $< x >$ représente la moyenne de $x$. D'où

$$det(\mathbf{X'X}) = N^2 s^2(x)$$

$(\mathbf{X'X})^{-1}$ est obtenue à partir de la matrice des cofacteurs :

$$(\mathbf{X'X})^{-\mathbf{1}} = \frac{1}{N^2 s^2(x)} \begin{pmatrix} c & -b \\ -b & a \end{pmatrix}$$

Nous voulons calculer les leviers $h_{ii} = \mathbf{x}_i^{'}(\mathbf{X'X})^{-1}\mathbf{x}_i$, donc :

$$h_{ii} = \frac{1}{N^2 s^2(x)} \begin{pmatrix} 1 & x_i \end{pmatrix} \begin{pmatrix} c & -b \\ -b & a \end{pmatrix} \begin{pmatrix} 1 \\ x_i \end{pmatrix}$$

d'où

$$h_{ii} = \frac{ax_i^2 - 2bx_i + c}{N^2 s^2(x)}$$

Les coefficients $a$,$b$ et $c$ sont fixés par le problème et nous voyons donc que les leviers varient en fonction de $x_i$ suivant un polynôme du second degré dont le minimum est atteint en :

$$\frac{b}{a} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Ce minimum correspond au centre de gravité de la distribution des points $x_i$. La valeur des leviers suit donc une parabole dont le minimum est atteint au centre de gravité des $x_i$.

En utilisant des variables centrées réduites $\tilde{x} = \frac{x - < x >}{\sigma_x}$, où $< x >$ et $\sigma_x$ sont la moyenne et l'écart type de $x$, l'équation de la parabole devient :

$$h_{ii} = \frac{1}{N}(\tilde{x}_i^2 + 1) \tag{2.17}$$

Pour un plan d'expériences de 6 points répartis comme dans la figure 2.3, nous obtenons les leviers de la figure 2.4.

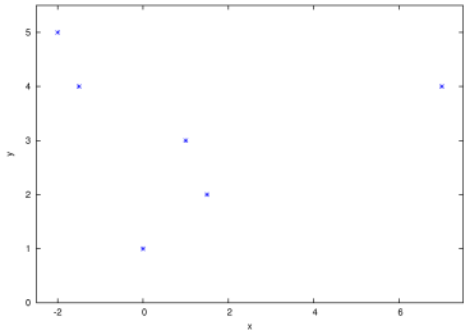



FIG. 2.3 – Répartition de 6 points

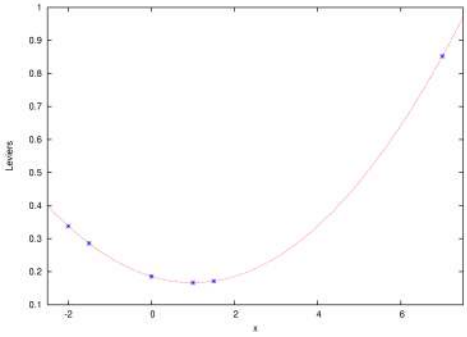



FIG. 2.4 – Les leviers se situent sur une parabole qui atteint son minimum en $x = 1$ qui est la moyenne des $x$

De manière générale, pour un modèle linéaire par rapport aux paramètres et par rapport aux facteurs, les leviers se situent sur un paraboloïde dont le minimum est atteint au centre de gravité des $\mathbf{x}_i$. Autrement dit, pour un tel modèle, plus un point est éloigné du centre de gravité de la distribution des points, plus il est influent sur l'estimation des paramètres.

Nous utiliserons les leviers dans le cadre de modèles polynomiaux, et les étendrons à des modèles non linéaires par rapport aux paramètres (réseaux de neurones) dans la suite de ce mémoire.

### 2.3.3 Le calcul des leviers

Le calcul de la matrice de projection et des leviers peut être mené simplement par une décomposition en valeurs singulières.

Soit $\mathbf{X}_{n\times p}$ une matrice rectangulaire avec $n > p$ de rang $r = p$. Il existe deux matrices orthogonales $\mathbf{U}_{n\times r}$ et $\mathbf{V}_{r\times r}$ ($\mathbf{U'U} = \mathbf{I}_r, \mathbf{V'V} = \mathbf{I}_r$), et une matrice diagonale $\mathbf{S} = \texttt{diag}(\lambda_1, \lambda_2, \ldots, \lambda_r), \lambda_i > 0$ telles que

$$\mathbf{X} = \mathbf{USV'}$$

Le carré des valeurs singulières correspond aux valeurs propres de la matrice $\mathbf{X'X}$.

La décomposition en valeurs singulières appliquées à la matrice d'expériences $\mathbf{X}$ donne :

$$\begin{aligned}
\mathbf{X} &= \mathbf{USV'} \texttt{ plan d'expérience} \\
(\mathbf{X'X}) &= \mathbf{VS^2V'} \texttt{ matrice d'information} \\
(\mathbf{X'X})^{-1} &= \mathbf{VS^{-2}V'} \texttt{ matrice de dispersion} \\
\mathbf{X}(\mathbf{X'X})^{-1}\mathbf{X'} &= \mathbf{UU'} = \mathbf{H} \texttt{ Projecteur}
\end{aligned}$$

En notant $\mathbf{u_i}$ la ligne $i$ de la matrice $\mathbf{U}$, les leviers $h_{ii}$, termes diagonaux de la matrice de projection $\mathbf{H} = \mathbf{UU'}$, vérifient donc :

$$h_{ii} = ||\mathbf{u_i}||^2$$

### 2.3.4 Propriétés de la matrice de projection H

Voici quelques propriétés concernant la matrice de projection $\mathbf{H}$.

D'après [Antoniadis *et al.* 92] :
Soit $\mathbf{X}_{N\times p}$ une matrice réelle, de rang $p$ et $\mathbf{H}$ la matrice de projection associée à la matrice $\mathbf{X}$. Alors :

- $\sum_{i=1}^{N} h_{ii} = p$,
- $\sum_{i=1}^{N}\sum_{j=1}^{N} h_{ij}^2 = p$,
- $0 \leq h_{ii} \leq 1$ pour tout i,
- $-0.5 \leq h_{ij} \leq 0.5$ quel que soit $j \neq i$,
- si $h_{ii} = 1$, ou $h_{ii} = 0$, alors $h_{ij} = 0$ pour $j \neq i$,
- $(1 - h_{ii})(1 - h_{jj}) - h_{ij}^2 \geq 0$,
- $h_{ii}h_{jj} - h_{ij}^2 \geq 0$.

De plus, si $\mathbf{X}$ a une colonne constante de composantes égales à 1, alors :

$$h_{ii} \geq \frac{1}{N}$$

C'est le cas dans l'ensemble de ce mémoire car la matrice $\mathbf{X}$ contient toujours une colonne de composantes égales à 1 (terme constant des modèles affines ou polynomiaux, terme de "biais" des réseaux de neurones). D'ailleurs, l'équation (2.17) montre, pour un modèle linéaire en ses paramètres, que le minimum de la parabole support des leviers vaut $1/N$.

## 2.4 Le Bootstrap

Comme nous l'avons indiqué dans l'introduction générale, nous utilisons la technique de ré-échantillonnage du Bootstrap pour estimer, de manière statistique, l'importance d'un exemple dans le processus d'apprentissage [Gazut & Martinez 04]. Nous allons, dans un premier temps, décrire cette technique de ré-échantillonnage avant de présenter quelques limites de cette méthode lorsqu'on l'utilise dans le cadre de la régression. Nous expliquerons, enfin, comment nous utiliserons le bootstrap dans ce mémoire.

Le bootstrap est une méthode de ré-échantillonnage qui existe sous plusieurs formes. Nous allons décrire, dans cette section, le bootstrap des individus.

### 2.4.1 Le Principe

Introduisons l'intérêt du Bootstrap en prenant l'exemple courant de la moyenne. Rappelons que la moyenne est la statistique $< X >= 1/N \sum_{i=1}^{N} X_i$ où les $X_i$ sont des variables aléatoires indépendantes et de même loi. En notant respectivement $m, \sigma^2$ l'espérance mathématique et la variance des variables $X_i$, l'espérance mathématique de la moyenne est $m$ (la moyenne est un estimateur non biaisé de l'espérance mathématique), et sa variance est $V(< X >) = \sigma^2/N$.

$$\begin{aligned} < X > &= \frac{1}{N} \sum_{i=1}^{N} X_i \\ E(< X >) &= m \\ V(< X >) &= \frac{\sigma^2}{N-1} \end{aligned}$$

Lorsque la variance $\sigma^2$ n'est pas connue, elle peut être estimée par la variance empirique de l'échantillon notée $s^2$

$$s^2 = \frac{1}{N} \sum_{i=1}^{N} (X_i - < X >)^2$$

La variance empirique de l'échantillon $s^2$ est une estimation biaisée de la variance $\sigma^2$ : $E(s^2) = \sigma^2(N-1)/N$ [Saporta 90]. La variance de la moyenne $< X >$ est alors estimée par :

$$\hat{V}(< X >) = \frac{1}{N(N-1)} \sum_{i=1}^{N} (X_i - < X >)^2$$

Pour la moyenne on a donc une formule estimant la variance. Mais pour de nombreuses autres statistiques comme par exemple la médiane, nous ne disposons pas de formules permettant d'estimer leur variance.

Pour estimer la variance d'une statistique, on peut utiliser le Bootstrap, technique de ré-échantillonnage, fondée sur la simulation d'échantillons à partir de l'échantillon initial. La validation croisée ou le leave one out sont aussi des techniques de ré-échantillonnage.

Le Bootstrap a été proposé par Efron en 1979 [Efron 79], pour une description détaillée et complète voir [Efron & Tibshirani 93]. Son principe est de ré-échantillonner $B$ fois un échantillon initial représentatif d'une population. Nous obtenons alors $B$ nouveaux échantillons appelés répliques. Ces répliques contiennent, en général, autant d'individus que l'échantillon initial et sont obtenues par tirage aléatoire avec remise des éléments de celui-ci. Le tirage se fait avec remise afin d'assurer l'indépendance des tirages. On augmente ainsi le nombre d'échantillons à partir des éléments du premier échantillon pour établir la statistique d'un estimateur.

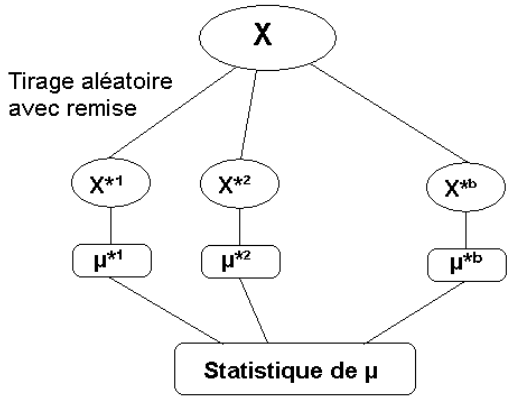



Fig. 2.5 – Schéma du bootstrap établissant une statistique d'un estimateur $\mu$

Par Bootstrap, la simulation de plusieurs échantillons permet d'observer plusieurs réalisations d'une statistique, la médiane empirique par exemple. Un échantillon, contenant autant de réalisations de la statistique que de répliques réalisées, sera donc constitué. On peut ainsi déterminer, pour la médiane empirique notée $\mu$, sa moyenne $m_{\mathcal{B}}$ et sa variance $s^2_{\mathcal{B}}$, de la manière suivante :

$$\begin{aligned} m_{\mathcal{B}} &= \frac{1}{B}\sum_{b=1}^{B}\mu^{*b} \\ s^2_{\mathcal{B}}(\mu) &= \frac{1}{B}\sum_{b=1}^{B}(\mu^{*b} - m_{\mathcal{B}})^2 \end{aligned}$$

où $\mu^{*b}$ est la valeur de la médiane pour la réplique $b$.

Pour la moyenne, il a été démontré dans [Efron & Tibshirani 93] que lorsque $B$ tend vers l'infini, la variance bootstrap tend vers la variance empirique. Évidemment, ne pouvant

réaliser une somme infinie pour l'obtenir, il faut choisir un nombre de répliques $B$ ; en pratique, on considère que cent ou deux cents répliques suffisent pour obtenir une bonne estimation de la moyenne ou de la variance.

On peut également établir une estimation de l'intervalle de confiance à 95 % pour $\mu$ en ne prenant que 95 % des valeurs centrales de la distribution bootstrap de $\mu$ [Wonnacott & Wonnacott 95]. Il est impossible d'avoir une telle statistique à partir du seul échantillon initial, d'autant plus que nous n'avons aucune hypothèse sur la distribution des individus dans l'échantillon initial qui nous aurait permis d'utiliser les méthodes paramétriques usuelles d'inférence statistique. C'est là que réside l'intérêt de cette méthode : elle ne nécessite aucune hypothèse concernant la distribution de l'échantillon initial $X$.

En raison du tirage aléatoire avec remise, un individu peut ne pas apparaître dans une réplique ou, au contraire, apparaître plusieurs fois. La probabilité qu'un exemple apparaisse $k$ fois dans une réplique de $N$ éléments suit la loi binomiale $B(k, p = \frac{1}{N}) = P(k) = C_N^k p^k (1-p)^{N-k}$ [Saporta 90]. La probabilité qu'un individu n'apparaisse pas dans une réplique est $P(0) = (1 - \frac{1}{N})^N$. La figure 2.6 représente l'évolution de cette probabilité en fonction du nombre $N$ d'individus ($lim_{N\to\infty} P(0) = e^{-1} \simeq 0.37$). Nous pouvons remarquer que la convergence est rapide et que pour une réplique contenant au moins une vingtaine d'exemples, la probabilité pour qu'un exemple n'apparaisse pas vaut environ $0,37$, proche de la valeur exacte $e^{-1}$.

Par la suite, nous analyserons le degré d'occurences des répliques en fonction du nombre d'exemples différents qu'elles contiennent, c'est-à-dire le dénombrement de la quantité de répliques contenant $k$ exemples différents parmi l'ensemble de toutes les répliques possibles de $N$ exemples. En effet, dans le cadre de la régression linéaire où le vecteur des parmètres est solution d'un système linéaire, le nombre d'exemples différents doit être supérieur au nombre de degré de liberté des modèles. On doit donc s'assurer que le nombre d'exemples différents contenus dans une réplique est supérieur au nombre de paramètres du modèle que l'on construit.

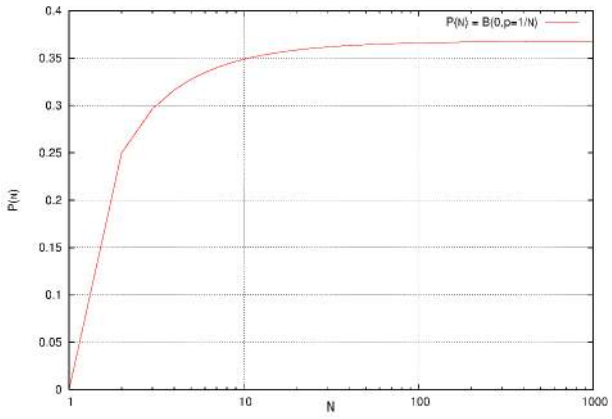



Fig. 2.6 – Probabilité qu'un individu n'apparaisse pas dans une réplique pour différentes valeurs du cardinal de la réplique $N$.

Dans le cadre de notre étude, nous utilisons le bootstrap pour la régression, avec deux objectifs :

– d'une part, afin de déterminer l'importance d'un exemple sur l'estimation des paramètres

d'un modèle,
– d'autre part, afin d'estimer l'erreur de généralisation d'un modèle.

Le premier de ces points sera abordé dans la section suivante ; le second sera traité dans le chapitre 3 sur la sélection de modèle.

### 2.4.2 Le Bootstrap en régression

En régression, la technique de ré-échantillonnage du Bootstrap nous intéresse pour deux raisons :

– elle ne nécessite aucune hypothèse sur les lois de distribution des résidus et sur celle des réponses prédites. Elle nous permettra donc d'estimer la stabilité de la construction du modèle apprenti par rapport à l'échantillon en analysant, par exemple, la variance des résidus, des réponses ou des paramètres du modèle apprenti.

– un lien intéressant peut être fait avec la notion de leviers, comme dans [Grandvalet 00]. En effet la méthode permet une estimation de la contribution relative de l'exemple à la construction du modèle apprenti.

Le schéma de la figure 2.7 illustre l'utilisation du Bootstrap en régression. On effectue $B$ apprentissages sur $B$ échantillons bootstrap. Les échantillons étant différents entre eux [5] nous avons $B$ modèles différents ayant chacun été créés par apprentissage d'une partie des exemples de la base initiale. Cette technique de ré-échantillonnage permet donc de réaliser une analyse statistique même dans les cas, comme le nôtre, où les expériences sont parfaitement déterministes puisqu'elles correspondent à des simulations numériques réalisées par des codes déterministes.

### 2.4.3 Analyse combinatoire

Comme nous l'avons précisé, il faut, en régression, que le nombre d'exemples différents soit supérieur (ou égal) au nombre de degré de liberté du modèle ; nous analysons, dans cette section, la probabilité du nombre d'exemples différents contenus dans une réplique.

Nous allons donc calculer la probabilité d'avoir $k$ exemples différents dans une réplique bootstrap générée à partir de la base d'exemples $\mathcal{L} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$. La réplique est réalisée par $N$ tirages avec remise sur $\mathcal{L}$. On note $i_1$ le nombre d'occurences de $\mathbf{x}_1$ dans la réplique, $i_2$ celui associé à $\mathbf{x}_2$ et ainsi de suite jusqu'à $i_N$ nombre d'occurences de $\mathbf{x}_N$. On a évidemment $\sum_{j=1}^{N} i_j = N$ puisque $N$ tirages ont été effectués.

La probabilité $P(i_1, i_2, \ldots, i_N)$ de réaliser une réplique définie par les occurences $i_j, j = 1, \ldots, N$, est donnée par la loi multinomiale $\mathcal{M}(N; p_1 = N^{-1}, \ldots, p_N = N^{-1})$ :

$$P(i_1, i_2, \ldots, i_N) = \frac{N!}{i_1! i_2! \ldots i_N!} \frac{1}{N^N} \tag{2.18}$$

[5] La probabilité d'avoir deux échantillons identiques est $\frac{1}{N^N}$ où $N$ est la nombre d'exemples de la base initiale.

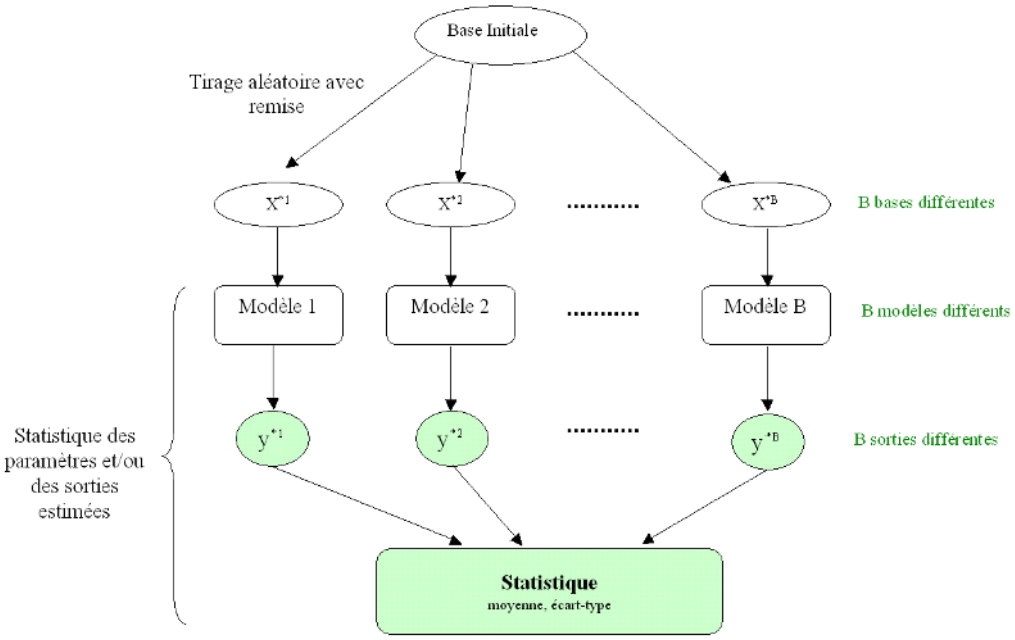



FIG. 2.7 – Utilisation de Bootstrap en régression

On appelle $(i_1, i_2, \ldots, i_N)$ l'indice multiple. Lorsque la réplique contient $k$ exemples différents, il y aura $k$ composantes de l'indice multiple différentes de 0. Pour le calcul de la probabilité d'occurrence d'une réplique contenant $k$ exemples différents, il suffit d'énumérer l'ensemble des occurrences $(i_1, i_2, \ldots, i_k)$ telles que $\sum_{j=1}^{k} i_j = N$.

Nous avons effectuer cette énumération sous forme algorithmique. En notant $\mathbf{I}_{\texttt{condition}}$ la fonction indicatrice qui vaut 1 si la `condition` est vérifiée et 0 sinon. Les indices multiples $(i_1, i_2, \ldots, i_k)$ tels que $\sum_{j=1}^{k} i_j = N$ peuvent être obtenus par :

$$\begin{aligned}
\sum_{i_j \geq 1, i_1+i_2+\ldots+i_k=N} 1 &= \sum_{i_1=1}^{N} \sum_{i_2=1}^{N} \cdots \sum_{i_k=1}^{N} \mathbf{I}_{i_1+i_2+\ldots+i_k=N} \\
&= \sum_{i_1=1}^{N-k+1} \sum_{i_2=1}^{N-k+2-i_1} \cdots \sum_{i_k=1}^{N-i_1-i_2\ldots-i_{k-1}} \mathbf{I}_{i_1+i_2+\ldots+i_k=N} \\
&= \sum_{i_1=1}^{N-k+1} \sum_{i_2=1}^{N-k+2-i_1} \cdots \sum_{i_k=N-i_1-i_2\ldots-i_{k-1}}^{N-i_1-i_2\ldots-i_{k-1}} 1
\end{aligned}$$

Cette énumeration correspond aussi à la dimension de l'espace des polynômes à $k$ variables de degré $N$ qui vaut $C_{N+k-1}^{N}$ [RAVIART & THOMAS 83].

$$\sum_{i_j \geq 1, i_1+i_2+\ldots+i_k=N} 1 = C_{N+k-1}^{N}$$

Pour un sous ensemble de $k$ exemples différents, cette énumération permet d'identifier le nombre de répliques contenant $k$ exemples différents. Chacune de ces répliques a une probabilité d'occurence définie par l'équation 2.18.

Pour obtenir la probabilité $P_N(k)$ d'obtenir une réplique contenant un sous ensemble quelconque de $k$ exemples pris parmi $N$, il faut prendre en considération le nombre de combinaisons possibles de constituer $k$ exemples pris parmi $N$, égal à $C_N^k$. On obtient alors la probabilité $P_N(k)$ d'obtenir une réplique contenant exactement $k$ exemples différents :

$$P_N(k) = \frac{C_N^k N!}{N^N} \sum_{i_1=1}^{N-k+1} \sum_{i_2=1}^{N-k+2-i_1} \cdots \sum_{i_j=1}^{N-k+j-i_1-i_2\ldots-i_{j-1}} \cdots \sum_{i_k=N-i_1-i_2\ldots-i_{k-1}}^{N-i_1-i_2\ldots-i_{k-1}} \frac{1}{i_1! i_2! \ldots i_k!} \quad (2.19)$$

La figure 2.8 représente cette fonction de probabilité pour $N = 100$.

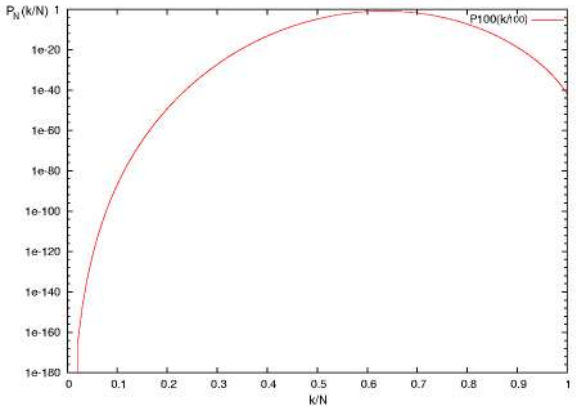



FIG. 2.8 – Probabilité $P(k/N)$ d'avoir une fraction $k/N$ d'exemples différents (calculée pour $N = 100$). En abscisse la valeur de la fraction $k/N$, en ordonnée le logarithme de $P_N(k/N)$

L'axe des abscisses représente le taux d'exemples différents, c'est-à-dire le rapport $\frac{k}{N}$, où $k$ représente le nombre d'exemples différents et $N$ le nombre total d'exemples contenu dans une réplique. L'axe des ordonnées représente la probabilité d'obtenir une réplique avec le taux d'exemples différents décrit en abscisse. Nous pouvons remarquer que le taux d'exemples pour lequel nous obtenons la plus grande probabilité d'apparition est 63 % d'exemples différents.

Nous pouvons estimer qu'en dessous d'un certain seuil, cette probabilité est nulle. La figure 2.9 représente cette même fonction de probabilité en échelle linéaire, et pour différentes valeurs de $N$.

L'évenement le plus probable est l'obtention d'une réplique contenant 63% des exemples de la base initiale. D'autre part, on observe une certaine symétrie de la densité de probabilité autour de la valeur maximale 63% qui correspond donc également à la valeur moyenne. L'écart autour de cette valeur de 63% diminue en fonction du cardinal de la base initiale. Nous pouvons visualiser la fonction de répartition pour différentes valeurs de $N$ (figure 2.10).

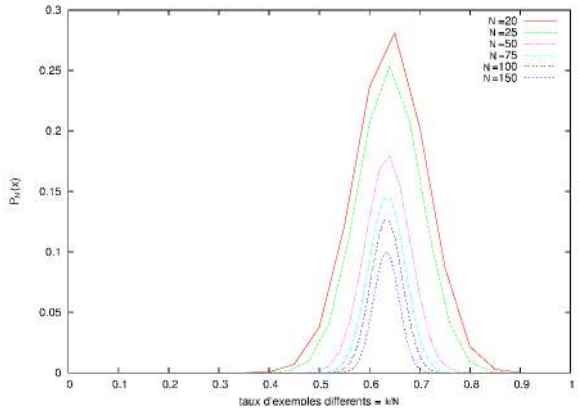



FIG. 2.9 – Bootstrap : probabilité d'avoir $k$ exemples distincts pour différentes valeurs de la taille de l'échantillon initial : $N = 20, 25, 50, 75, 100, 150$

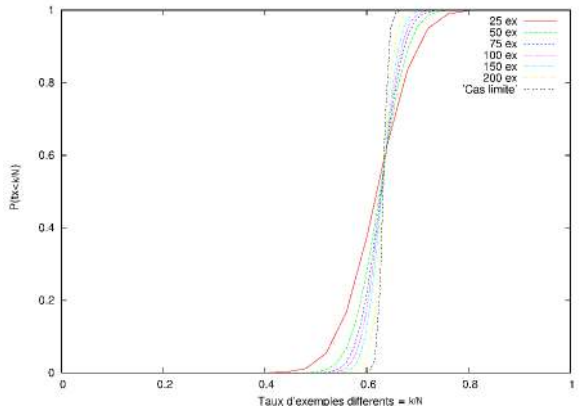



FIG. 2.10 – Bootstrap : fonction de répartition du taux $k/N$ d'exemples distincts pour différentes valeurs de la taille de l'échantillon initial : $N = 25, 50, 75, 100, 150, 200$. Il faut garder à l'esprit que $k/N$ n'est pas une variable continue et ne peut prendre que $N$ valeurs entre 0 et 1.

Toute cette étude de probabilité nous permet de déterminer le nombre d'exemples différents contenu dans une réplique de $N$ expériences, nous pouvons alors savoir combien de paramètres nous pourrons ajuster sur celles-ci et donc, connaître la complexité maximale des modèles qu'elles pourront entraîner. Il faut lire la figure 2.10 comme la probabilité d'obtenir une réplique dont le taux d'exemples différents est inférieur au taux d'exemples différents représenté en abscisse. Supposons, par exemple, que nous ayons $N = 50$ et que l'on veuille vérifier si un modèle à 10 paramètres peut être entraîné. Il nous faut donc au minimum 10 exemples différents soit un taux de 0.2. La probabilité d'avoir une réplique dont le taux d'exemples différents est inférieur à 0.2 est quasiment nulle. On aura donc, presque à coup sûr, au moins 10 exemples différents sur chacune des répliques. Les apprentissages des modèles sur celles-ci se feront convenablement et la statistique bootstrap sur ces modèles sera alors robuste.

S'il y a trop peu d'exemples pour l'architecture utilisée, nous préconisons d'utiliser la validation croisée ou, dans le cas limite, le leave-one-out pour établir des statistiques similaires.

### 2.4.4 Le Leave-Many-Out

Nous avons étudié, au paragraphe sur le leave-one-out (page 48), l'influence du retrait d'un seul exemple sur l'estimation des paramètres d'un modèle mais lorsque l'on utilise le Bootstrap c'est en moyenne 37 % des exemples que l'on retire simultanément. On ne peut donc pas utiliser directement la formule du leave-one-out pour déterminer la variance des paramètres. Il faudrait donc déterminer l'influence du retrait d'une famille d'exemples sur l'estimation des paramètres du modèle. Par analogie au leave-one-out, nous avons appelé "formule du leave-many-out" la formule que nous allons établir ci-dessous.

Notons par $\mathbf{X}_{(a,b)}$ la matrice des effets dont nous avons extrait les exemples $a$ et $b$, ce qui correspond à ôter la ligne $\boldsymbol{\phi}(\mathbf{x}_b)'$ de la matrice $\mathbf{X}_{(a)}$ qui est la matrice des effets dont nous avons extrait l'exemple $a$.

$$\mathbf{X}'_{(a,b)}\mathbf{X}_{(a,b)} = \mathbf{X}'_{(a)}\mathbf{X}_{(a)} - \boldsymbol{\phi}(\mathbf{x}_b)\boldsymbol{\phi}(\mathbf{x}_b)'$$

En utilisant le lemme d'inversion (2.5) pour $\mathbf{A} = \mathbf{X}'_{(a)}\mathbf{X}_{(a)}$, $\mathbf{a} = -\boldsymbol{\phi}(\mathbf{x}_b)$ et $\mathbf{b} = \boldsymbol{\phi}(\mathbf{x}_b)$.
Nous obtenons l'expression de $(\mathbf{X}'_{(a,b)}\mathbf{X}_{(a,b)})^{-1}$ en fonction de $(\mathbf{X}'_{(a)}\mathbf{X}_{(a)})^{-1}$.

$$(\mathbf{X}'_{(a,b)}\mathbf{X}_{(a,b)})^{-1} = (\mathbf{X}'_{(a)}\mathbf{X}_{(a)})^{-1} + \frac{(\mathbf{X}'_{(a)}\mathbf{X}_{(a)})^{-1}\boldsymbol{\phi}(\mathbf{x}_b)\boldsymbol{\phi}(\mathbf{x}_b)'(\mathbf{X}'_{(a)}\mathbf{X}_{(a)})^{-1}}{1 - \boldsymbol{\phi}(\mathbf{x}_b)'(\mathbf{X}'_{(a)}\mathbf{X}_{(a)})^{-1}\boldsymbol{\phi}(\mathbf{x}_b)}$$

en posant $h_{bb}^{(a)} = \boldsymbol{\phi}(\mathbf{x}_b)'(\mathbf{X}'_{(a)}\mathbf{X}_{(a)})^{-1}\boldsymbol{\phi}(\mathbf{x}_b)$ le levier de l'exemple $b$, élément diagonal $b$ de la matrice chapeau calculée sans l'exemple $a$, nous obtenons l'équation :

$$(\mathbf{X}'_{(a,b)}\mathbf{X}_{(a,b)})^{-1} = (\mathbf{X}'_{(a)}\mathbf{X}_{(a)})^{-1} + \frac{(\mathbf{X}'_{(a)}\mathbf{X}_{(a)})^{-1}\boldsymbol{\phi}(\mathbf{x}_b)\boldsymbol{\phi}(\mathbf{x}_b)'(\mathbf{X}'_{(a)}\mathbf{X}_{(a)})^{-1}}{1 - h_{bb}^{(a)}} \tag{2.20}$$

En conduisant les mêmes calculs que pour le leave-one-out de la page 48 à partir de l'équation (2.20), on obtient la formule du leave-many-out pour le retrait des deux exemples $a$ et $b$ :

$$r_j^{(a,b)} = r_j^{(a)} + \frac{h_{jb}^{(a)} r_b^{(a)}}{1 - h_{bb}^{(a)}} \tag{2.21}$$

On peut généraliser cette équation en nommant $\mathcal{D}$ une liste de retrait d'exemples. Bien entendu, $Card(\mathcal{D}) < N$ où $N$ est le nombre d'exemples.

Pour $a \notin \mathcal{D}$ :

$$r_j^{(\mathcal{D},a)} = r_j^{(\mathcal{D})} + \frac{h_{aj}^{(\mathcal{D})}}{1 - h_{aa}^{(\mathcal{D})}} \times r_a^{(\mathcal{D})} \tag{2.22}$$

Il est finalement difficile d'utiliser directement cette équation car elle nécessite un nombre important d'évaluations de matrices chapeau en tenant compte des exemples contenus ou non dans les répliques. Nous avons donc utiliser la méthode classique d'estimation par bootstrap telle que décrite précédemment au paragraphe 2.4.2.

## 2.5 Relation entre les leviers et la variance bootstrap

Nous allons illustrer les paragraphes suivants par l'utilisation d'un exemple "jouet" : le sinus cardinal. Cet exemple nous permettra de visualiser, en dimension 1 et 2 les profils de variance et de leviers que nous obtenons pour les bases considérées. La méthode que nous décrivons, et que nous validons dans le cadre linéaire, sera utilisée dans la suite du mémoire sur des modèles non linéaires.

Nous commencerons par rappeler les règles classiques du modèle probabiliste en régression. On suppose les erreurs aléatoires, non corrélés, de moyenne nulle et de variance $\sigma^2$. Les paramètres estimés sont alors des variables aléatoires. Rappelons l'expression de leur matrice de covariance.

La solution des moindres carrés est

$$\boldsymbol{\theta}_{\mathcal{L}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$

d'où la variance des paramètres estimés

$$s^2(\boldsymbol{\theta}_{\mathcal{L}}) = s^2((\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}) = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}' s^2(\mathbf{y})\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1} = \sigma^2(\mathbf{X}'\mathbf{X})^{-1} \tag{2.23}$$

On peut déterminer, également, la variance des sorties estimées : les paramètres étant des variables aléatoires, les sorties estimées le sont aussi.

$$s^2(f(\mathbf{x}_i, \boldsymbol{\theta}_{\mathcal{L}})) = s^2(\boldsymbol{\phi}(\mathbf{x}_i)'\boldsymbol{\theta}_{\mathcal{L}}) = \boldsymbol{\phi}(\mathbf{x_i})' s^2(\boldsymbol{\theta}_{\mathcal{L}})\boldsymbol{\phi}(\mathbf{x}_i) = \boldsymbol{\phi}(\mathbf{x_i})'(\mathbf{X'X})^{-1}\boldsymbol{\phi}(\mathbf{x}_i)\sigma^2 = h_{ii}\sigma^2 \quad (2.24)$$

Sous les hypothèses du modèle probabiliste de la régression, la variance des sorties estimées par le modèle, pour l'exemple $\mathbf{x}_i$, vaut donc le produit du levier de l'exemple $i$ et de la variance du bruit.

Dans le cas déterministe, c'est-à-dire dans le cas où les erreurs expérimentales ne sont dûes qu'aux erreurs d'approximation, la relation entre le levier et la variance $\sigma^2$ ne peut plus être appliquée telle quelle dans la mesure où la variance $\sigma^2$ n'existe pas. Nous allons voir dans les paragraphes qui suivent que nous pourrons, par bootstrap, reconstituer cette relation.

Pour l'ensemble des tests qui vont suivre, nous allons utiliser une base d'apprentissage dont le générateur est la fonction sinus cardinal $\frac{sin(x)}{x}$ sur l'intervalle [-4 ;10]. La base d'apprentissage est constituée de 20 points répartis uniformément. Nous n'avons pas ajouté de bruit puisque nous nous plaçons dans le cadre d'expériences simulées à partir d'un code de calcul déterministe.

Nous allons créer, dans un premier temps, des modèles linéaires en leurs paramètres et linéaires par rapport aux facteurs, puis des modèles polynomiaux en fin de chapitre. Nous allons donc faire une régression linéaire sur cette base d'exemples. Il peut paraître aberrant d'approcher un sinus cardinal par une droite, mais déjà, dans cette configuration, nous pouvons montrer des résultats intéressants. D'autre part, il est très facile, dans un cas simple à un facteur, comme celui-ci, de déceler une inadéquation entre les expériences et le modèle ; pour des dimensions supérieures, on peut très souvent approcher une fonction analytique inconnue par un modèle qui n'aurait pas une complexité suffisante. C'est d'ailleurs tout le problème du dilemme biais/variance vu au chapitre 1. Ce cas particulier correspond à l'un des aspects de ce problème : le cas du biais important et de la variance faible.

Nous pouvons voir à la figure 2.11, la fonction génératrice sinus cardinal, les points d'expériences ainsi que le modèle obtenu. Nous allons distinguer deux cas : celui d'un modèle linéaire pour lequel on n'estime qu'un seul paramètre, et celui d'un modèle affine pour lequel on estime deux paramètres.

Dans ces deux cas d'étude, nous allons tenter d'estimer la variance des paramètres par rapport aux exemples par Bootstrap. Il faut souligner que dans le cas déterministe, nous aurons toujours les mêmes paramètres de régression si nous reconduisons l'apprentissage à partir de la même spécification des exemples. Comme nous l'avons précisé au paragraphe 2.4.2, c'est le tirage bootstrap qui nous permet, même dans le cadre de données déterministes, de considérer les paramètres estimés comme des variables aléatoires et de calculer leur variance.

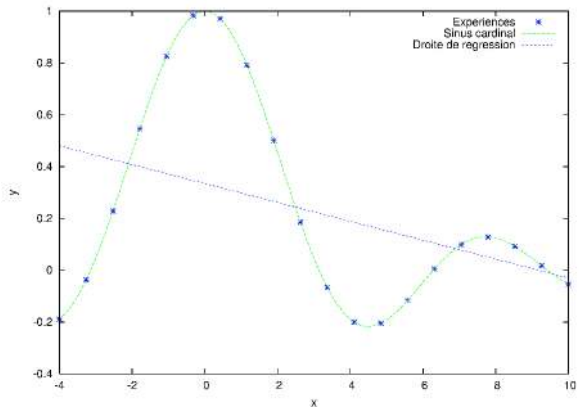



FIG. 2.11 – Points de la base d'apprentissage de la fonction sinus cardinal et le modèle obtenu.

### 2.5.1 Pour un modèle linéaire

On se place dans le cas d'un modèle linéaire $f(x,\theta) = \theta_1 x$. Nous effectuons, avant l'apprentissage, une normalisation des variables et des sorties afin de ramener celles-ci à des ordres de grandeurs comparables. Posons $\tilde{y} = \frac{y - <y>}{\sigma_y}$ et $\tilde{x} = \frac{x - <x>}{\sigma_x}$, les variables centrées réduites des $x$ et $y$, où $< y >$ et $\sigma_y$ sont respectivement la moyenne et l'écart-type de l'échantillon des $y_i$ et $< x >$ et $\sigma_x$ la moyenne et l'écart-type des $x_i$.

Dans ce paragraphe nous cherchons donc le paramètre $\theta_1$ solution des moindres carrés dans l'espace normalisé tel que :

$$f(\tilde{x},\theta) = \theta_1 \tilde{x} \tag{2.25}$$

Nous cherchons à exprimer la variance bootstrap des prédictions du modèle en $\tilde{x}$ :

$$s^2_{\mathcal{B}}(f(\tilde{x},\theta)) = s^2_{\mathcal{B}}(\theta_1 \tilde{x}) \tag{2.26}$$

Nous avons autant d'estimations du paramètre $\theta_1$ que nous avons de répliques puisque chacun des échantillons bootstrap est différent. L'estimation des paramètres est donc différente et par conséquent la variance $s^2_{\mathcal{B}}(\theta_1)$ est non nulle. La figure 2.12 représente les modèles obtenus à partir des 200 répliques bootstrap dans l'espace normalisé.

D'après l'équation (2.26), la variance des prédictions du modèle en $\tilde{x}$ s'exprime de la manière suivante :

$$s^2_{\mathcal{B}}(f(\tilde{x},\theta)) = s^2_{\mathcal{B}}(\theta_1)\tilde{x}^2$$

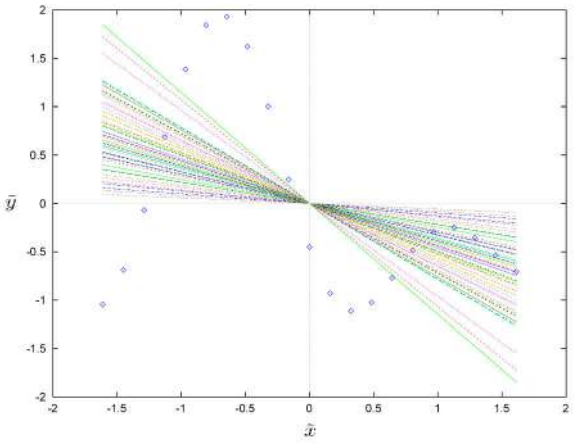



FIG. 2.12 – Modèles obtenus à partir des 200 répliques bootstrap dans l'espace normalisé

Donc $s^2_{\mathcal{B}}(f(\tilde{x},\theta))$ varie comme le carré de $\tilde{x}$. Or, d'après la relation (2.17), les leviers varient comme :

$$h_{ii} = \frac{\tilde{x}_i^2 + 1}{N} \tag{2.27}$$

La variance bootstrap des prédictions peut donc s'exprimer en fonction des leviers :

$$s^2_{\mathcal{B}}(f(\tilde{x},\theta)) = N s^2_{\mathcal{B}}(\theta_1) h_{ii} - s^2_{\mathcal{B}}(\theta_1)$$

Les 200 modèles obtenus étant linéaires, la variance des prédictions à l'origine est nulle, comme nous pouvons le voir à la figure 2.12. Dans l'espace de départ, ces 200 modèles linéaires deviennent des modèles affines dont le terme constant est obtenu à partir des paramètres de normalisation.

$$\begin{aligned} \frac{f(x,\theta) - < y >}{\sigma_y} &= \theta_1 \frac{x - < x >}{\sigma_x} \\ f(x,\theta) &= \theta_1 \frac{\sigma_y}{\sigma_x} x + < y > - \theta_1 \frac{\sigma_y}{\sigma_x} < x > \end{aligned}$$

La figure 2.13 représente les 200 modèles dans l'espace de départ.

Le point de variance de prédiction nulle est alors obtenu au centre de gravité des $x_i$. C'est en ce point que l'on observe le minimum des leviers. La figure 2.14 illustre la relation entre la variance des prédictions et les leviers. Cette relation s'explique simplement. Si une réplique ne contient pas d'exemples situés sur les bords du domaine, l'absence de ces exemples, à levier élevé,

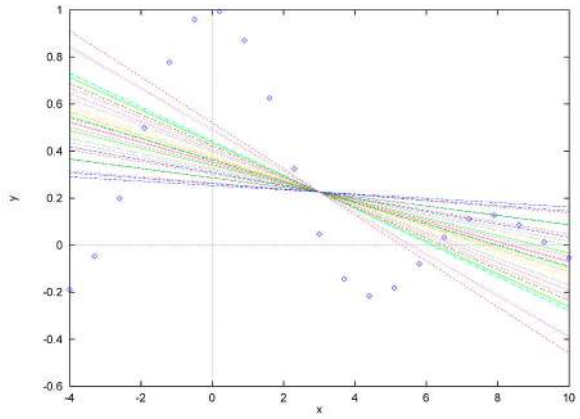



FIG. 2.13 – Modèles obtenus à partir des 200 répliques bootstrap dans l'espace de départ

provoque une forte variation des estimations des paramètres. De même, les points proches du centre de gravité (à levier faible) n'ont pas une grande influence sur l'estimation des paramètres.

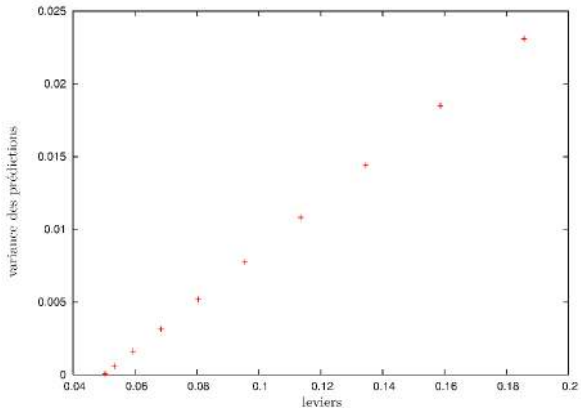



FIG. 2.14 – Corrélation entre la variance des prédictions par Bootstrap et les leviers pour un processus déterministe

Le choix d'approcher le sinus cardinal par des modèles linéaires dans l'espace normalisé a eu pour conséquence d'imposer le point de variance minimale (en l'occurence de variance nulle) à l'origine de l'espace normalisé. Ce point de variance nulle correspond, dans l'espace de départ, au centre de gravité du nuage de points de la base d'apprentissage. Cette procédure est justifiée lorsque l'on effectue un seul apprentissage en utilisant toutes les expériences. Le terme constant du modèle affine dans l'espace de départ est alors obtenu à partir des paramètres de normalisation. Par contre, dans le cadre de l'utilisation que l'on fait du bootstrap, ce n'est plus justifié. Le paramètre $\theta_1$ est estimé, dans ce cas, avec seulement 63 % des exemples alors que le terme constant est obtenu à partir des paramètres de normalisation calculés avec l'ensemble des exemples. Il est donc préférable d'estimer également le biais pour chacune des répliques et d'obtenir ainsi, comme pour le paramètre $\theta_1$, la variance bootstrap du terme constant.

### 2.5.2 Pour un modèle affine

Cette fois-ci le terme constant $\theta_0$ est estimé durant l'apprentissage de chacun des $B$ modèles dont la base d'apprentissage est une réplique bootstrap. Nous aurons donc $B$ valeurs des paramètres $(\theta_0, \theta_1)$.

Sous ces conditions :

$$f(\tilde{x}, \boldsymbol{\theta}) = \theta_0 + \theta_1 \tilde{x}$$

donc

$$s^2_{\mathcal{B}}(f(\tilde{x}, \boldsymbol{\theta})) = s^2_{\mathcal{B}}(\theta_0 + \theta_1 \tilde{x}) \quad (2.28)$$

$$s^2_{\mathcal{B}}(f(\tilde{x}, \boldsymbol{\theta})) = s^2_{\mathcal{B}}(\theta_1)\tilde{x}^2 + 2covar_{\mathcal{B}}(\theta_0, \theta_1)\tilde{x} + s^2_{\mathcal{B}}(\theta_0) \quad (2.29)$$

La variance de la prédiction est donc encore quadratique en $\tilde{x}$, mais son minimum n'est plus situé au centre de gravité des $x_i$. La position de l'axe de symétrie de la parabole dépend de $covar_{\mathcal{B}}(\theta_0, \theta_1)$.

La figure 2.15 représente l'allure de cette parabole pour la base d'apprentissage décrite en début de paragraphe.

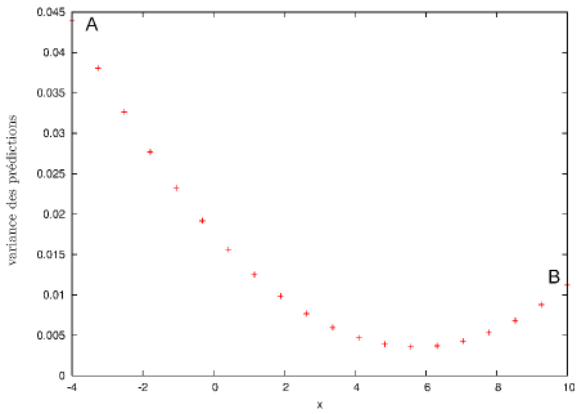



FIG. 2.15 – Répartition de la variance bootstrap des prédictions pour $covar_{\mathcal{B}}(\theta_0, \theta_1) < 0$.

Le fait d'estimer le terme constant $\theta_0$ à chaque réplique est intéressant puisqu'il nous donne une information sur la zone où la sortie du générateur d'exemple (le sinus cardinal) a de fortes variabilités. La variance bootstrap des prédictions nous donne alors une information plus riche que celle donnée par les leviers. Dans le cadre linéaire, les leviers désignent un point comme étant influent sur les paramètres du modèle s'il est loin du centre de gravité des $\mathbf{x}_i$ sans jamais prendre en compte l'information sur les sorties $y$ associées aux exemples $\mathbf{x}$. La variance bootstrap des prédictions donne de l'importance d'une part, aux expériences au bord du domaine, mais aussi aux exemples situés dans les zones à forte variabilité suivant les $y$.

En effet, lorsque l'on part d'un tirage initial uniforme, les points sont répartis régulièrement sur le domaine. Lorsque les points situés dans une zone à forte variabilité n'apparaissent pas dans une réplique, les paramètres, notamment le biais, sont fortement modifiés. Pour le sinus cardinal, la zone de plus forte variabilité de la sortie se situe entre $x = -4$ et $x = 4$. Un point peut donc avoir de l'influence d'une part s'il est loin du centre du domaine, mais également s'il appartient à une zone de forte variabilité [6].

Pour illustrer ceci, nous avons visualisé les 200 modèles bootstrap (voir figure 2.16) qui ont conduit à estimer la variance des prédictions de la figure 2.15 :

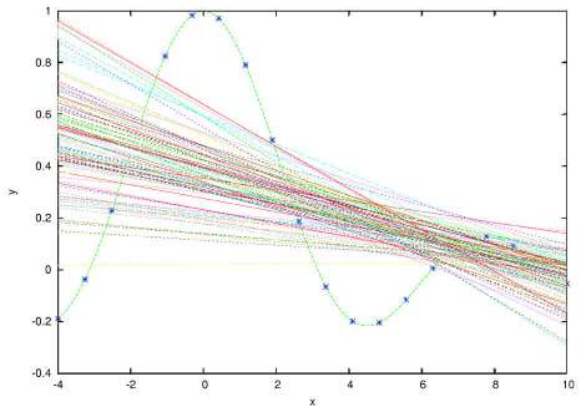



FIG. 2.16 – 200 modèles construits à partir de répliques bootstrap. On retrouve le minimum de la variance des estimations localisé en $x \simeq 6$ par la Figure 2.15.

Lorsque l'on estime le biais pour chacune des répliques bootstrap, la variance bootstrap des prédictions ne varie pas linéairement avec les leviers (voir (2.28) et (2.27)) :

$$s^2_{\mathcal{B}}(f(\tilde{x}_i, \boldsymbol{\theta})) = s^2_{\mathcal{B}}(\theta_1)(Nh_{ii} - 1) \pm 2cov_{\mathcal{B}}(\theta_0, \theta_1)\sqrt{Nh_{ii} - 1} + s^2_{\mathcal{B}}(\theta_0)$$

La figure 2.17 représente la variance bootstrap des prédictions en fonction des leviers.

Deux exemples symétriques par rapport au centre de gravité des exemples ont des leviers égaux, mais des valeurs de variance bootstrap différentes. Pour différentes bases d'expériences, nous avons noté par A et B les points au bord du domaine. Comme le centre de gravité de la distribution de points est au centre du domaine, les leviers des points A et B sont identiques dans tous les cas. En revanche, la variance bootstrap des prédictions en ces points est différente suivant la position de la zone de forte variabilité de la fonction sinus cardinal.

La figure 2.18 montre les profils de variance bootstrap des prédictions pour différentes bases d'apprentissage obtenues par translation du sinus cardinal. Les fortes valeurs de variance bootstrap des prédictions s'orientent vers les zones de fortes variabilité de $y$.

[6] Ceci rappel les travaux menés dans [ANTONIADIS *et al.* 92] qui proposent de prendre en compte l'information de sortie dans le calcul des leviers. Nous allons en parler brièvement à la fin de ce chapitre.

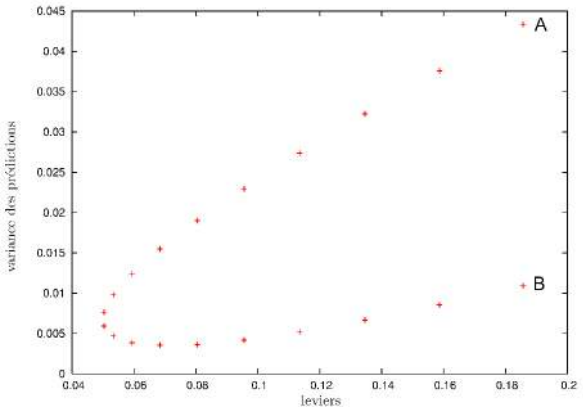



FIG. 2.17 – Relation entre la variance bootstrap des prédictions et les leviers pour un modèle déterministe pour lequel on estime le terme constant.

### 2.5.3 Généralisation en dimension deux

L'étude que nous avons faite à une dimension peut être étendue facilement aux dimensions supérieures. Nous allons montrer, dans ce paragraphe, l'extension à deux facteurs afin de pouvoir visualiser facilement les résultats avant de passer aux modèles polynomiaux en fin de chapitre. La généralisation pour des espaces de dimension élevée ne pose pas de problème majeur. Nous utilisons la surface obtenue à partir de la fonction sinus cardinal à 2 facteurs ($\frac{sin(\sqrt{x_1^2+x_2^2})}{\sqrt{x_1^2+x_2^2}}$) dans le domaine $[-4;10]^2$ et un maillage de points centrés sur lesquels seront évaluées les variances de prédictions. Même si nous avons souligné précédemment qu'il fallait estimer le biais pour chacune des répliques bootstrap, nous allons faire tout d'abord une étude pour un modèle linéaire sans biais[7] afin de pouvoir comparer avec les résultats précédents.

#### Modèle Linéaire

Nous avons un paramètre de plus à estimer que dans le cas précédent :

$$\begin{aligned} f(\tilde{\mathbf{x}}, \boldsymbol{\theta}) &= \theta_1 \tilde{x}_1 + \theta_2 \tilde{x}_2 \\ s^2_{\mathcal{B}}(f(\tilde{\mathbf{x}}, \boldsymbol{\theta})) &= s^2_{\mathcal{B}}(\theta_1 \tilde{x_1} + \theta_2 \tilde{x_2}) \end{aligned}$$

d'où

$$s^2_{\mathcal{B}}(f(\tilde{\mathbf{x}}, \boldsymbol{\theta})) = s^2_{\mathcal{B}}(\theta_1)\tilde{x}_1^2 + 2\tilde{x}_1\tilde{x}_2 cov_{\mathcal{B}}(\theta_1, \theta_2) + s^2_{\mathcal{B}}(\theta_2)\tilde{x_2}^2 \tag{2.30}$$

Etant donné que la base d'apprentissage est centrée, les leviers sont distribués sur un paraboloïde dont le minimum est atteint au centre de gravité des $\mathbf{x}_i$ qui est aussi le centre du domaine. Soit $h(x_1, x_2)$ la valeur du levier du point $(\tilde{x_1}, \tilde{x_2})$.

$$h(\tilde{x}_1, \tilde{x}_2) = a\tilde{x}_1^2 + b\tilde{x}_2^2 + \frac{1}{N}$$

[7]Le biais étant obtenu après dénormalisation des données.

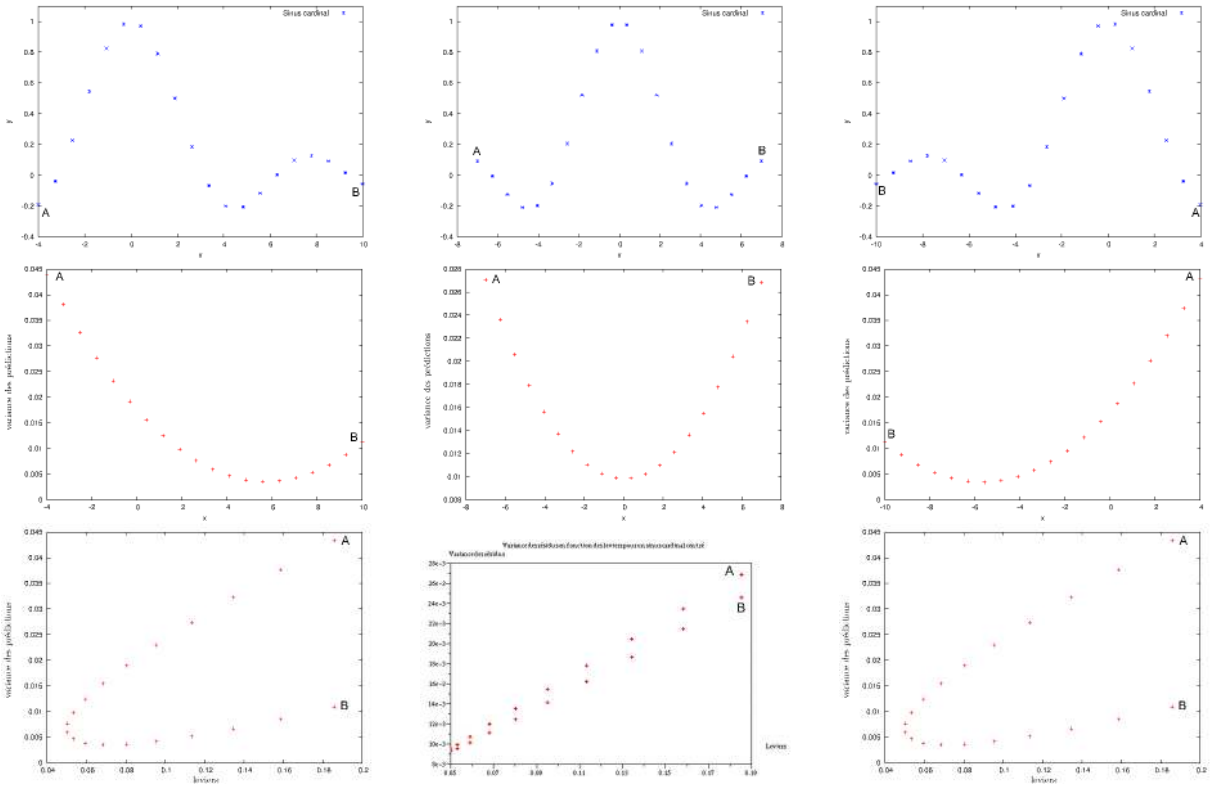


FIG. 2.18 – Relation entre la variance bootstrap des prédictions et les leviers pour différentes valeurs de cov($\theta_0$,$\theta_1$)

La valeur $\frac{1}{N}$ du terme constant correspond à la valeur du levier du centre de gravité des $\mathbf{x}$ (voir le paragraphe 2.3.4). La figure 2.20 montre les lignes de niveau de la surface décrite par les leviers (courbes iso-leviers). Le maillage étant régulier, $a$ et $b$ sont égaux, donc les courbes iso-leviers sont circulaires.

L'équation (2.30) montre que, les courbes d'isovariance bootstrap sont de forme elliptique puisque, $s^2_{\mathcal{B}}(\theta_1) \neq s^2_{\mathcal{B}}(\theta_2)$. De plus, la covariance entre $\theta_1$ et $\theta_2$ étant non nulle, il existe un terme croisé : les axes de symétrie de ces ellipses ne sont pas parallèles aux axes $x_1$ et $x_2$. La figure 2.21 représente les lignes de niveau pour les surfaces décrites par les leviers et la variance bootstrap.

Sur la figure 2.21, nous pouvons voir effectivement que les lignes de niveau d'isovariance sont elliptiques et donc que, **contrairement au cas des leviers, la distribution de la variance bootstrap n'est pas isotropique ; certaines directions sont plus importantes que d'autres dans le cadre de la variance bootstrap.** La figure 2.22 montre la relation entre la variance bootstrap des prédictions et les leviers.

Nous obtenons deux droites enveloppes correspondant aux corrélations entre la variance bootstrap et les leviers sur les axes de symétries des courbes elliptiques d'isovariance, et des points répartis de manière parabolique entre ces deux droites du fait du maillage. Les enveloppes sont des droites car le modèle n'est pas affine. Comme dans le cas à une dimension, le choix du modèle linéaire impose une variance bootstrap nulle au centre de gravité des $\mathbf{x}_i$.

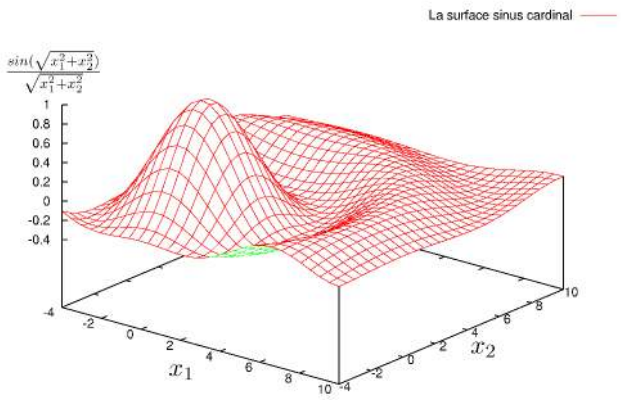



FIG. 2.19 – La surface sinus cardinal.

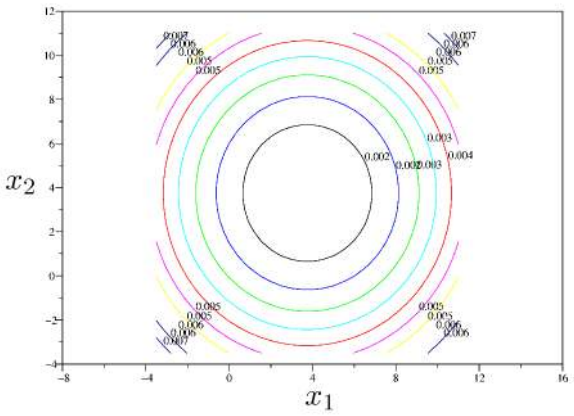



FIG. 2.20 – Lignes de niveau de la surface décrite par les leviers.

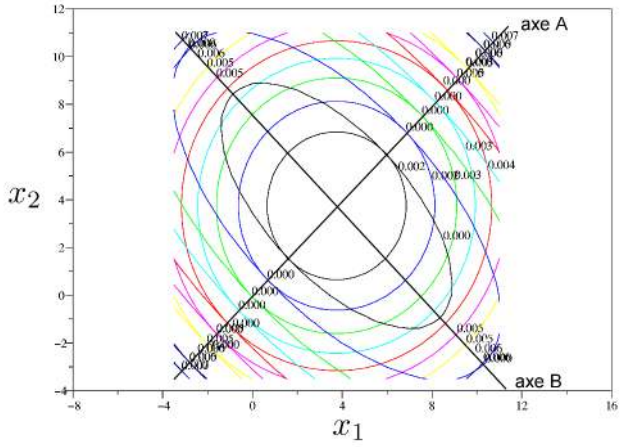



FIG. 2.21 – Lignes de niveau de la surface décrite par les leviers et celle de la variance bootstrap.

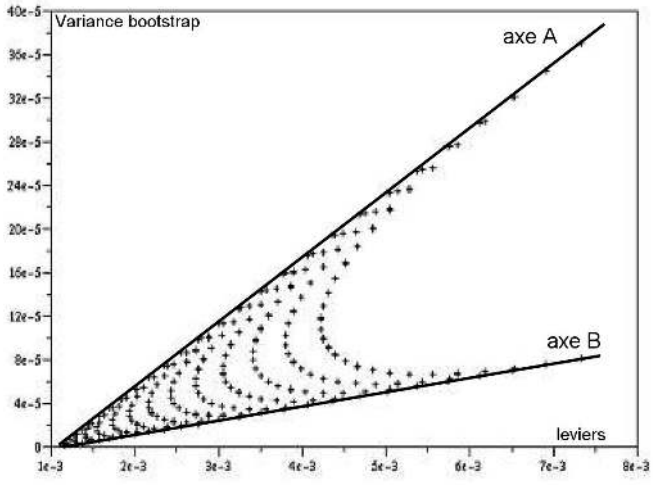



FIG. 2.22 – Relation entre la variance bootstrap et les leviers pour un modèle linéaire en dimension 2.

**Modèle affine**

Comme dans le cas mono-dimensionnel, nous étudions l'effet de l'estimation du teme constant du modèle sur la variance bootstrap.

Nous estimons la variance des sorties estimées à partir du modèle suivant :

$$\begin{aligned} f(\tilde{\mathbf{x}}, \boldsymbol{\theta}) &= \theta_0 + \theta_1 \tilde{x}_1 + \theta_2 \tilde{x}_2 \\ s^2_{\mathcal{B}}(f(\tilde{\mathbf{x}}, \boldsymbol{\theta})) &= s^2_{\mathcal{B}}(\theta_0 + \theta_1 \tilde{x}_1 + \theta_2 \tilde{x}_2) \end{aligned}$$

$$s^2_{\mathcal{B}}(f(\tilde{\mathbf{x}}, \boldsymbol{\theta})) = s^2_{\mathcal{B}}(\theta_1)\tilde{x}_1^2 + s^2_{\mathcal{B}}(\theta_2)\tilde{x}_2^2 + 2cov_{\mathcal{B}}(\theta_1, \theta_2)\tilde{x}_1\tilde{x}_2 + 2cov_{\mathcal{B}}(\theta_0, \theta_1)\tilde{x}_1 + 2cov_{\mathcal{B}}(\theta_0, \theta_2)\tilde{x}_2 + s^2_{\mathcal{B}}(\theta_0) \tag{2.31}$$

La répartition des leviers est la même que dans les cas précédents car nous n'avons pas changé la base d'exemples, mais pour la variance bootstrap des prédictions, en plus du terme croisé, nous voyons que le minimum ne sera pas atteint au centre du domaine car nous avons des termes non nuls devant $\tilde{x}_1$ et $\tilde{x}_2$. La figure 2.23 représente les lignes de niveau pour les leviers et la variance bootstrap pour un modèle affine.

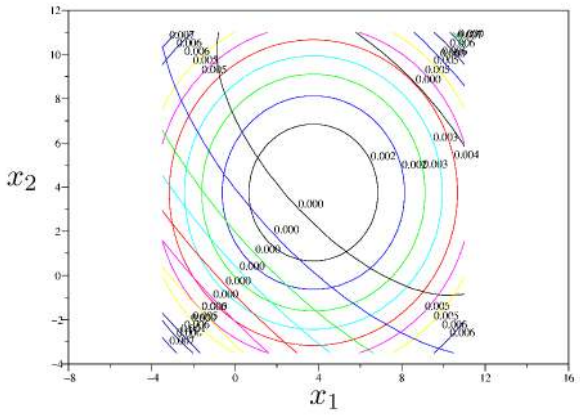



FIG. 2.23 – Lignes de niveau de la surface décrite par les leviers et celle de la variance bootstrap dans le cas d'un modèle affine.

Nous voyons très nettement (figure 2.23) l'apparition à la fois du terme croisé et des monômes $\tilde{x_1}$ et $\tilde{x_2}$.

Finalement **les valeurs de variance bootstrap sont les plus fortes dans la zone où la variabilité de la fonction sinus cardinal l'est également (zone** $[-4;4]^2$**).** Lorsque les exemples de cette zone ne font pas partie d'une réplique, les paramètres estimés du plan affine sont fortement modifiés, d'où une forte variabilité sur la variance bootstrap des sorties estimées. Avant de passer à des modèles plus en adéquation avec la fonction sinus cardinal, nous allons voir l'exemple du sinus cardinal sur le domaine $[-7;7]^2$. Sur ce domaine, la fonction sinus cardinal est centrée, le maximum de la fonction est alors atteint au centre du domaine d'étude.

### Sinus cardinal centré

La figure 2.24 représente la fonction sinus cardinal dans le domaine $[-7:7]^2$.

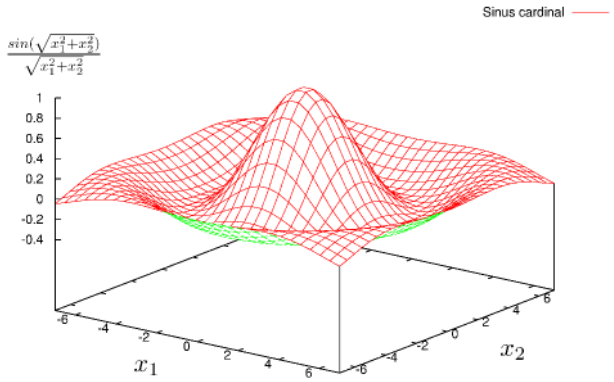



FIG. 2.24 – Le sinus cardinal centré.

Dans ce cas, les plans affines estimés sur chacune des répliques sont quasi-horizontaux. Ce "quasi" dépend du tirage bootstrap des points de la base initiale. Les points dans la zone de forte variabilité du sinus cardinal modifient surtout l'estimation du terme constant. La covariance des paramètres est très faible du fait que l'on a un plan horizontal. La surface décrite par la variance bootstrap est alors très voisine de celle décrite par les leviers : paraboloïde, dont le minimum est atteint au centre du domaine d'étude. La figure 2.25 représente les lignes de niveau des leviers et de la variance bootstrap dans ce cas.

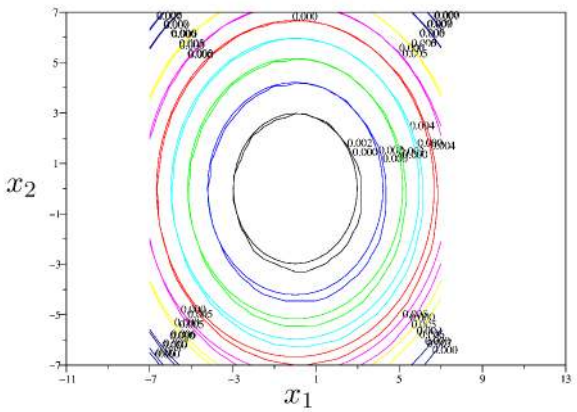



FIG. 2.25 – Projection sur le même plan des leviers et de la variance bootstrap.

La variance bootstrap et les leviers sont alors très corrélés comme le montre la figure 2.26.

Ici, la variance bootstrap des sorties estimées, ne nous renseigne pas sur les zones du sinus cardinal à forte variabilité, car il ne faut pas oublier que le modèle que l'on génère est un plan

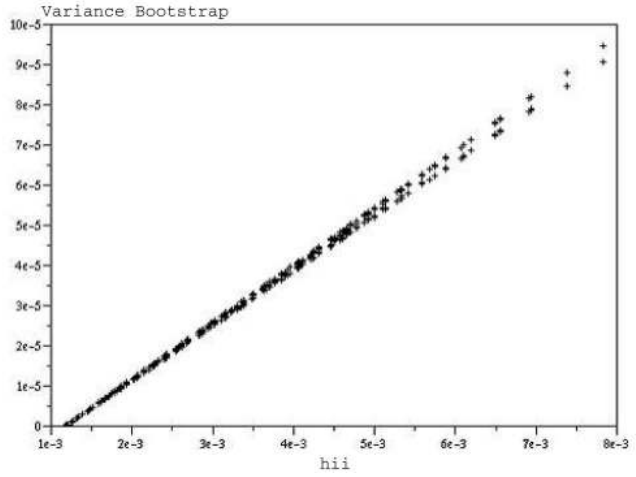



FIG. 2.26 – Corrélation entre la variance bootstrap et les leviers pour le sinus cardinal centré.

et qu'il ne permet pas de saisir les symétries du problème. Nous verrons dans le paragraphe suivant que, dans le cadre de la modélisation polynomiale, on pourra détecter cette zone de forte variabilité.

### 2.5.4 La variance bootstrap pour des modèles polynomiaux

Nous avons consacré deux paragraphes à illustrer l'intérêt de la variance bootstrap pour estimer l'importance d'un exemple, mais, comme nous l'avions précisé, les modèles linéaires en leurs paramètres et en leurs facteurs ne sont pas du tout appropriés pour approcher le sinus cardinal même si, dans les cas réels en dimension élevée, on ne sait pas toujours si le modèle choisi *a priori* est en adéquation avec le problème. Dans ce paragraphe, nous cherchons à approcher le sinus cardinal par un polynôme de degré trois à deux facteurs.

#### Calcul de la variance des sorties estimées

Comme précédemment, $B$ répliques bootstrap sont générées à partir de la base d'exemples initiale. La méthode des moindres carrés est mise en oeuvre pour créer un modèle par réplique. Notons $\boldsymbol{\theta}_{\mathcal{L}^{*b}}$ le vecteur de paramètres estimés sur la réplique bootstrap $\mathcal{L}^{*b}$, dont les informations sont contenues dans la matrice $\mathbf{X}_b$ ; notons $\mathbf{y}_b$ le vecteur des sorties associé à $\mathbf{X}_b$.

$$\boldsymbol{\theta}_{\mathcal{L}^{*b}} = (\mathbf{X}_b'\mathbf{X}_b)^{-1}\mathbf{X}_b'\mathbf{y}_b$$

Soit $\mathbf{P}$ la matrice des paramètres estimés centrés ($\theta^i_{\mathcal{L}^{*b}}$ est le ième élément du vecteur des paramètres estimés sur la réplique bootstrap b).

$$P = \begin{pmatrix} \theta^1_{\mathcal{L}^{*1}} - <\theta^1> & \theta^2_{\mathcal{L}^{*1}} - <\theta^2> & \dots & \theta^p_{\mathcal{L}^{*1}} - <\theta^p> \\ \theta^1_{\mathcal{L}^{*2}} - <\theta^1> & \theta^2_{\mathcal{L}^{*2}} - <\theta^2> & \dots & \theta^p_{\mathcal{L}^{*2}} - <\theta^p> \\ \vdots & \vdots & \vdots & \vdots \\ \theta^1_{\mathcal{L}^{*B}} - <\theta^1> & \theta^2_{\mathcal{L}^{*B}} - <\theta^2> & \dots & \theta^p_{\mathcal{L}^{*B}} - <\theta^p> \end{pmatrix}$$

Cette matrice a $B$ lignes et $p$ colonnes, où $p$ est le nombre de paramètres du modèle. On a alors :

$$
\begin{aligned}
s^2_{\mathcal{B}}(\boldsymbol{\theta}) &= \frac{1}{B}\mathbf{P}'\mathbf{P} \\
\left[s^2_{\mathcal{B}}(\boldsymbol{\theta})\right]_{ij} &= \frac{1}{B}\sum_{k=1}^{B}(\theta^i_{\mathcal{L}^{*k}} - < \theta^i >)(\theta^j_{\mathcal{L}^{*k}} - < \theta^j >)
\end{aligned}
$$

Nous pouvons alors obtenir les variances de prédictions par bootstrap en différents points d'un maillage. Notons $\mathbf{M}$ la matrice des points du maillage qui contient autant de lignes que de points dans le maillage et autant de colonnes que de paramètres. On obtient :

$$
\begin{aligned}
s^2_{\mathcal{B}}(f(\mathbf{x}, \boldsymbol{\theta})) &= s^2(\mathbf{M}\boldsymbol{\theta}) \\
&= \mathbf{M}s^2_{\mathcal{B}}(\boldsymbol{\theta})\mathbf{M}' \\
&= \frac{1}{B}\mathbf{M}(\mathbf{P}'\mathbf{P})\mathbf{M}'
\end{aligned}
$$

La termes diagonaux de la matrice $s^2_{\mathcal{B}}(f(\mathbf{x}, \boldsymbol{\theta}))$ représentent les variances de prédictions estimées par bootstrap aux points du maillage.

### La variance bootstrap avec un modèle polynomial

Nous approchons la fonction sinus cardinal à deux facteurs par une surface polynomiale de degré trois. Nous utilisons 50 points obtenus par tirage LHS (Latin Hypercube Sampling) [Iman *et al.* 81] dans le domaine $[-4; 10]^2$ comme base d'apprentissage. Historiquement une grille carrée $n \times n$ est un carré latin si chaque modalité (parmi les $n$) du facteur interne (les deux autres facteurs définissant les $n$ lignes et les $n$ colonnes du carré latin) n'apparaît qu'une fois et une seule par ligne et par colonne. Un hypercube latin est la généralisation possible pour $p$ supérieur à trois facteurs, telle que chacun des $n$ p-uples n'apparaît qu'une fois et une seule. La méthode LHS permet donc de générer un tirage multidimensionnel, constituant une base d'apprentissage, en imposant des lois d'apparition pour chacune des variables suivant les propriétés géométriques de la grille. Par exemple, si l'on désire créer un tirage multidimensionel pour lequel chacune des variables suit une loi uniforme, il faut appliquer la méthode de l'hypercube latin à une grille régulière comme le montre la figure 2.27. Si l'on désire créer un tirage multidimensionnel pour lequel une variable est obtenue par un tirage uniforme, et une autre obtenue par un tirage gaussien, il faut utiliser une grille similaire à celle de la figure 2.28.

Le modèle que l'on utilise contient $p = 10$ paramètres. Le vecteur $\boldsymbol{\phi}(\mathbf{x}_i)$ des entrées est :

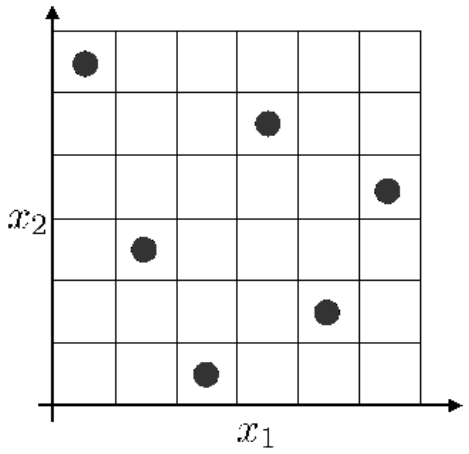



FIG. 2.27 – Tirage LHS pour lequel chacune des variables est tirée suivant une loi uniforme.

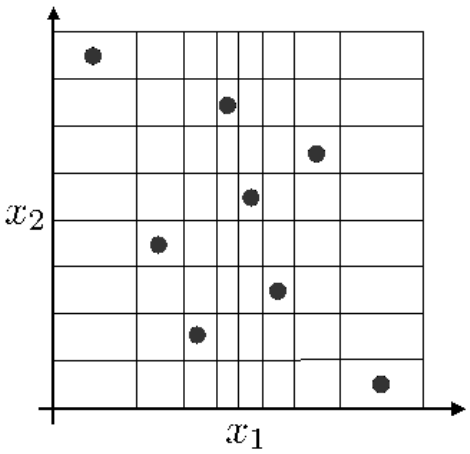



FIG. 2.28 – Tirage LHS pour lequel la variable $x_1$ est tirée suivant une loi normale et la variable $x_2$ tirée suivant une loi uniforme. Chaque cellule de la grille présente la même probabilité de tirage.

$$\boldsymbol{\phi}(\mathbf{x}_i) = \begin{pmatrix} \phi_0(\mathbf{x}_i) = 1, \\ \phi_1(\mathbf{x}_i) = x_i, \\ \phi_2(\mathbf{x}_i) = y_i, \\ \phi_3(\mathbf{x}_i) = x_i^2, \\ \phi_4(\mathbf{x}_i) = x_i y_i, \\ \phi_5(\mathbf{x}_i) = y_i^2, \\ \phi_6(\mathbf{x}_i) = x_i^3, \\ \phi_7(\mathbf{x}_i) = x_i^2 y_i, \\ \phi_8(\mathbf{x}_i) = x_i y_i^2, \\ \phi_9(\mathbf{x}_i) = y_i^3 \end{pmatrix}$$

La figure 2.29 montre la surface polynomiale de degré trois qui approche la fonction sinus cardinal obtenue par moindres carrés.

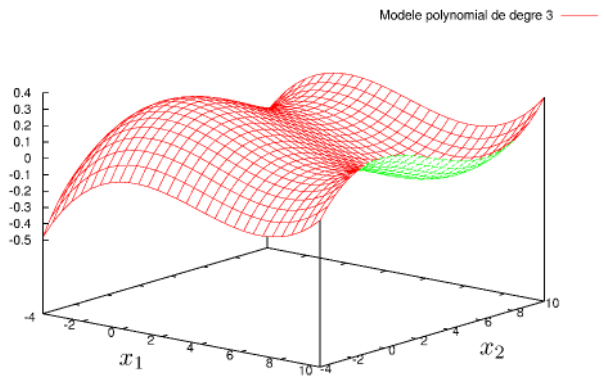



FIG. 2.29 – La surface polynomiale de degré 3 approchant le sinus cardinal.

Nous utilisons un maillage de 10000 points pour calculer les leviers, la variance bootstrap et visualiser les surfaces. La figure 2.30 représente la surface décrite par les leviers sur le domaine d'étude ; rappelons que $h_{ii} = \boldsymbol{\phi}(\mathbf{x}_i)'(\mathbf{X}'\mathbf{X})^{-1}\boldsymbol{\phi}(\mathbf{x}_i)$, où $\boldsymbol{\phi}(\mathbf{x}_i)$ est le vecteur des variables secondaires de l'exemple $i$.

La figure 2.31 représente les lignes de niveau de cette surface.

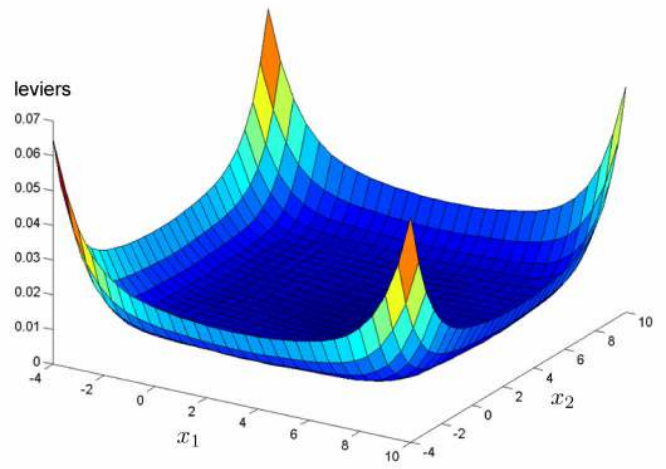



FIG. 2.30 – Les leviers pour la surface polynomiale de degré 3.

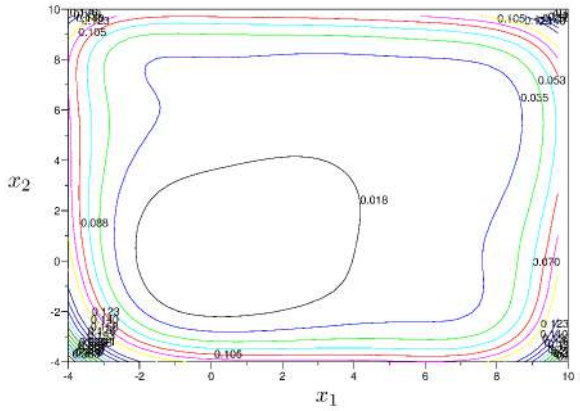



FIG. 2.31 – Lignes de niveau des leviers pour la surface polynomiale de degré 3.

La variance bootstrap des prédictions est représentée sur les figures 2.32 et 2.33.

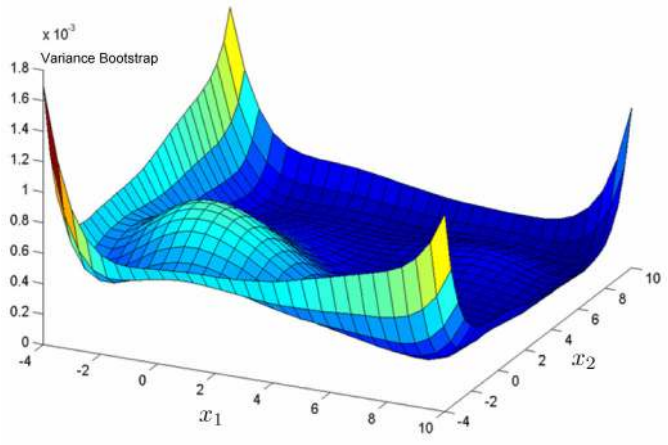



FIG. 2.32 – La variance bootstrap pour la surface polynomiale de degré 3.

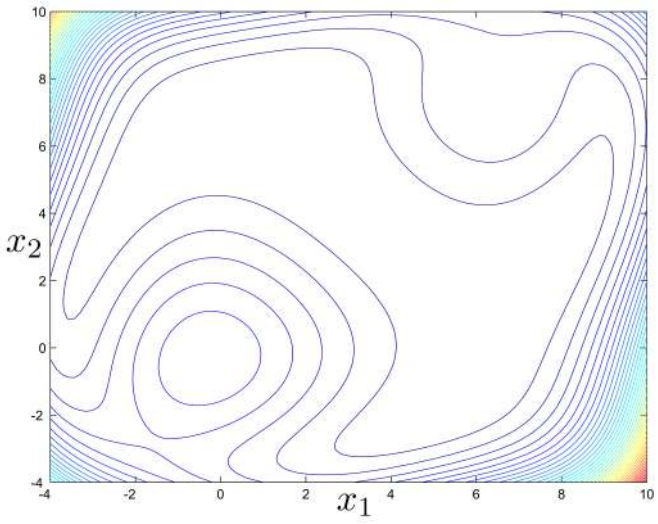



FIG. 2.33 – Lignes de niveau de la variance bootstrap pour la surface polynomiale de degré 3.

Nous voyons très nettement que la variance bootstrap des prédictions est sensible aux zones à fortes variations de la fonction génératrice à modéliser (le sinus cardinal). Dans ces zones la variance est plus importante. L'approche est intéressante car nous obtenons ce type d'information sur un maillage, uniquement avec les données disponibles dans la base d'apprentissage. Nous allons comparer dans le paragraphe suivant la variance bootstrap et une approche fondée sur des leviers calculés avec la sortie du processus.

### 2.5.5 Des leviers avec prise en compte de la sortie

Dans [ANTONIADIS *et al.* 92], les auteurs ajoutent le vecteur de sortie à la matrice des expériences $\mathbf{X}$ pour obtenir la matrice notée $\mathbf{W} = (\mathbf{X}, \mathbf{Y})$. Comme nous l'avons décrit plus tôt,

$\mathbf{H_X} = \mathbf{X}(\mathbf{X'X})^{-1}\mathbf{X'}$, nous aurons donc $\mathbf{H_W} = \mathbf{W}(\mathbf{W'W})^{-1}\mathbf{W'}$ ; on peut montrer que l'on a alors :

$$\mathbf{H_W} = \mathbf{H_X} + \frac{(\mathbf{I} - \mathbf{H_X})\mathbf{Y'Y}(\mathbf{I} - \mathbf{H_X})}{\mathbf{Y'}(\mathbf{I} - \mathbf{H_X})\mathbf{Y}} = \mathbf{H_X} + \frac{\boldsymbol{\epsilon}\boldsymbol{\epsilon'}}{\boldsymbol{\epsilon'}\boldsymbol{\epsilon}} \tag{2.32}$$

où $\boldsymbol{\epsilon}$ est le vecteur dont la composante $i$ vaut $\epsilon_i = y_i - f(\mathbf{x}_i, \boldsymbol{\theta}_{\mathcal{L}})$.

La statistique $h_{ii} + \frac{\epsilon_i^2}{\boldsymbol{\epsilon'}\boldsymbol{\epsilon}}$ apparaît comme une nouvelle mesure de l'influence d'un exemple sur la construction du modèle. En effet, cette quantité est grande lorsque le levier $h_{ii}$ est grand, mais aussi lorsque le résidu de l'exemple $i$ l'est également. Ces leviers qui prennent en considération la sortie mesurent donc à la fois l'influence de la $i^{\text{ème}}$ observation et l'inadéquation du modèle à cette observation.

La figure 2.34 représente la surface de ces leviers, pour le même maillage que précédemment. Les lignes de niveau correspondant à cette surface sont représentées sur la figure 2.35

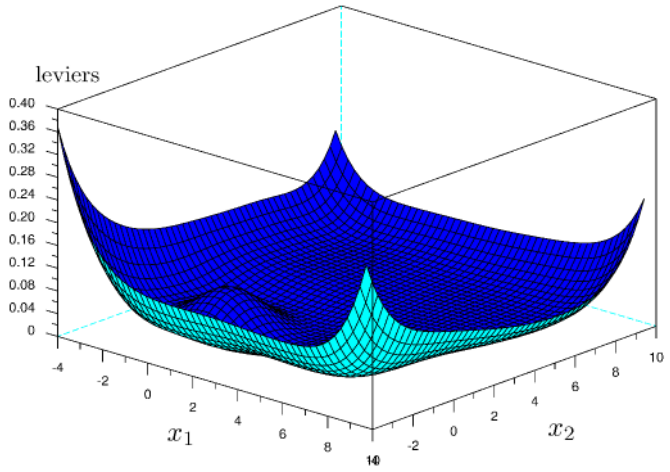



FIG. 2.34 – Surface décrite par les éléments diagonaux de la matrice $H_W$ pour la surface polynomiale de degré 3.

Ces leviers, qui prennent en compte l'information de sortie, permettent de retrouver, comme le fait la variance bootstrap, la zone proche de l'origine de forte variabilité où le modèle est mal ajusté. Cependant, pour obtenir ces résultats avec ces leviers, nous avons utilisé un maillage pour lequel nous avions l'information de sortie en chaque point, car sans le vecteur $\mathbf{y}$, le calcul des éléments diagonaux de la matrice $\mathbf{H_W}$ est impossible. Malheureusement, dans un cas réel, nous n'avons pas autant d'information et autant de points, et le calcul de ces leviers n'est alors possible que pour les points de la base d'apprentissage. La variance bootstrap des sorties estimées offre l'avantage d'obtenir ce type d'information uniquement avec les points de la base d'apprentissage. Autrement dit, nous avons une idée du manque d'ajustement du modèle (biais

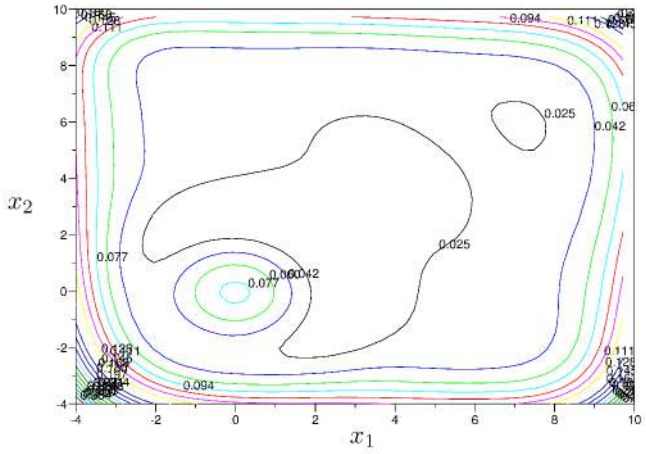



FIG. 2.35 – Lignes de niveau des éléments diagonaux de la matrice $H_W$ pour la surface polynomiale de degré 3.

important) sur l'ensemble du maillage grâce au ré-échantillonnage bootstrap.

## 2.6 Conclusion

Dans ce chapitre, les notions de levier et de variance bootstrap ont été introduites. Ces deux techniques cherchent, par des moyens différents, à mettre en évidence l'influence d'un exemple sur l'estimation des paramètres d'un modèle ; nous les avons étudiées en détail dans le cadre de la modélisation linéaire, et nous avons mis en évidence les ressemblances et différences entre elles, à l'aide d'un exemple jouet à 1 et 2 dimensions.

Avant d'aborder l'extension de ces méthodes au cas de la modélisation non linéaire, qui est l'objet de notre travail, nous étudierons, dans le chapitre suivant, le problème du choix de la complexité du modèle d'une part, et, d'autre part, le problème de la sélection de modèles de même complexité dans le cas où la fonction de coût qui est minimisée durant l'apprentissage présente des minima locaux.

# Chapitre 3

# Choix de modèles

**Sommaire**


**3.1 Introduction . . . . . 87**
**3.2 Estimation de l'erreur de généralisation dans le cadre linéaire . . . 87**
**3.3 Estimation de l'erreur de généralisation par des techniques de ré-échantillonnage . . . . . 93**
3.3.1 Validation croisée . . . . . 94
3.3.2 Le Leave-One-Out . . . . . 94
3.3.3 Estimation de l'erreur de généralisation par Bootstrap . . . . . 96
**3.4 Choix d'une complexité . . . . . 98**
3.4.1 Principe . . . . . 98
3.4.2 Quelques exemples . . . . . 100
**3.5 Heuristiques pour l'initialisation des paramètres et pour la régularisation par arrêt prématuré . . . . . 105**
3.5.1 Choix du meilleur modèle par leave-one-out virtuel pour une architecture donnée . . . . . 105
3.5.2 Cas particulier des processus déterministes . . . . . 106
3.5.3 Procédure utilisée pour la construction des modèles par bootstrap . . . 108
**3.6 Conclusion . . . . . 110**

## 3.1 Introduction

Nous abordons ici l'un des problèmes les plus importants en apprentissage, celui, encore ouvert, de l'estimation de l'erreur de généralisation. En effet, la méthode fondée sur la minimisation du risque structurel proposée par Vapnik, nécessite la connaissance de la dimension VC. Mais le calcul de la dimension VC reste, aujourd'hui, un problème ouvert pour la plupart des classes de fonctions utilisées en apprentissage statistique. Nous nous sommes donc orientés vers des méthodes statistiques pour estimer le risque fonctionnel (erreur de généralisation).

Dans ce chapitre nous présenterons, tout d'abord, une technique d'estimation de l'erreur de généralisation pour des modèles linéaires par rapport aux paramètres fondée sur une analyse spectrale de la matrice d'information $\mathbf{X}'\mathbf{X}$. Cette méthode ne sera pas utilisable directement pour des modèles non linéaires par rapport aux paramètres, c'est pourquoi nous présenterons des méthodes d'estimation de l'erreur de généralisation fondées sur des techniques de ré-échantillonnage comme le Leave-One-Out, la validation croisée et le Bootstrap. Ces méthodes peuvent être employées dans le cas où les fonctions de base de $\mathcal{F}$ utilisées par l'apprenti sont non linéaires par rapport aux paramètres comme, par exemple, les réseaux de neurones que nous utiliserons au dernier chapitre pour valider la méthode de planification d'expériences fondée sur l'apprentissage actif. Le problème soulevé par ce type de fonctions est dû au fait que le coût n'est pas convexe. Sa minimisation doit être contrôlée afin d'éviter les minima locaux. On s'est donc attaché à analyser les corrélations entre le minimum de l'erreur d'apprentissage (risque empirique) et celui de l'erreur de généralisation (risque fonctionnel). Nous rappellerons également la méthode présentée dans [Monari & Dreyfus 00] fondée sur l'utilisation, sous certaines hypothèses, du Leave-One-Out pour les fonctions non linéaires. On terminera ce chapitre par quelques exemples illustrant et comparant la validation croisée et le bootstrap dans le problème du choix optimal de la complexité du modèle.

## 3.2 Estimation de l'erreur de généralisation dans le cadre linéaire

Nous allons présenter dans cette partie la méthode d'estimation du risque fonctionnel fondée sur une analyse spectrale de la matrice d'information présentée dans [Chapelle 04] dans le domaine de l'apprentissage statistique.

On note $y(\mathbf{x})$ la fonction de régression du superviseur :

$$E(y|\mathbf{x}) = \int yP(y|\mathbf{x})dy = y(\mathbf{x})$$

La fonction de régression est la solution optimale minimisant le risque fonctionnel à coût quadratique. Retenons comme famille de régresseurs, une base de polynômes orthonormés notés $\phi_i$, associés à la densité de probabilité $P(\mathbf{x})$. Les fonctions $\phi_i$ vérifient la relation suivante :

$$\int \phi_i(\mathbf{x})\phi_j(\mathbf{x})P(\mathbf{x})d\mathbf{x} = \delta_{i,j}$$

$\delta_{i,j}$ est le symbole de Kronecker

$$\begin{aligned}\delta_{i,j} &= 1 \texttt{ si } i = j \\ &= 0 \texttt{ sinon}\end{aligned}$$

Le polynôme $\phi_0$ correspond à la fonction constante ($\forall \mathbf{x}, \phi_0(\mathbf{x}) = 1$). Les régresseurs $\phi_i, i > 0$ ont donc une moyenne nulle. La fonction de régression se développe sur la base des polynômes orthogonaux. En notant $\alpha_i$ les coefficients du développement :

$$y(\mathbf{x}) = \sum_{i=0}^{\infty} \alpha_i \phi_i(\mathbf{x})$$

Supposons que le modèle postulé soit une combinaison linéaire d'un nombre fini $p$ de polynômes de la base tronquée, de la forme :

$$f(\mathbf{x}, \boldsymbol{\theta}) = \sum_{i=0}^{p} \theta_i \phi_i(\mathbf{x})$$

Le risque fonctionnel est :

$$\begin{aligned}R(f) &= \int (f(\mathbf{x}, \boldsymbol{\theta}) - y(\mathbf{x}))^2 P(\mathbf{x}) d\mathbf{x} &(3.1)\\ &= \int [\sum_{i=0}^{p} (\theta_i - \alpha_i)\phi_i(\mathbf{x}) - \sum_{i=p+1}^{\infty} \alpha_i \phi_i(\mathbf{x})]^2 P(\mathbf{x}) d\mathbf{x} &(3.2)\end{aligned}$$

Compte tenu de l'orthogonalité et de la normalisation des régresseurs, le risque fonctionnel s'exprime en fonctions des coefficients $\alpha$ de la fonction de régression et des paramètres $\theta$ de l'apprenti. L'erreur se décompose donc en deux parties. La première correspond à l'erreur d'estimation due à l'écart entre les $p+1$ premiers coefficients. La seconde correspond à l'erreur d'approximation ou de troncature, qui est donc incompressible.

$$R(f) = \underbrace{\sum_{i=0}^{p} (\theta_i - \alpha_i)^2}_{\texttt{Erreur d'estimation}} + \underbrace{\sum_{i=p+1}^{\infty} \alpha_i^2}_{\texttt{Erreur d'approximation}} \quad (3.3)$$

Le meilleur apprenti est donc celui qui annule l'erreur d'estimation, c'est-à-dire l'apprenti $\sum_{i=0}^{p} \theta_i \phi_i(\mathbf{x})$ vérifiant $\theta_i = \alpha_i$ pour $i = 0, 1, \ldots, p$. Nous allons donc nous intéresser à l'erreur d'estimation.

Notons par $R_t(f)$ l'erreur d'estimation apparaissant dans le risque fonctionnel. Posons $\boldsymbol{\alpha}_t$ le vecteur des $\alpha$ tronqué c'est-à-dire $\boldsymbol{\alpha}_t = (\alpha_0, \alpha_1, \ldots, \alpha_p)'$. On désigne par $\mathbf{y}_t$ la projection de la fonction de régression sur les $p$ polynômes de la base de $\mathcal{F}$ et par $\mathbf{y}_u$ le reste lié à l'erreur de troncature.

$$\mathbf{y}(\mathbf{x}) = \underbrace{\sum_{i=0}^{p} \alpha_i \phi_i(\mathbf{x})}_{\mathbf{y}_t} + \underbrace{\sum_{i=p+1}^{\infty} \alpha_i \phi_i(\mathbf{x})}_{\mathbf{y}_u} \tag{3.4}$$

En reprenant les notations de la section précédente, et en décomposant le vecteur des réponses $\mathbf{y}$ en $\mathbf{y} = \mathbf{y}_t + \mathbf{y}_u = \mathbf{X}\boldsymbol{\alpha}_t + \mathbf{y}_u$, l'erreur d'estimation $R_t(f)$ se développe de la façon décrite à l'equation (3.5). $\theta_{i\mathcal{L}}$ correspond à la composante $i$ du vecteur $\boldsymbol{\theta}_{\mathcal{L}}$ qui est le vecteur de paramètres qui minimise la fonction de coût des moindres carrés. $\boldsymbol{\theta}_{\mathcal{L}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$.

$$\begin{aligned}
R_t(f) &= \sum_{i=0}^{p} (\theta_{i\mathcal{L}} - \alpha_i)^2 \\
&= (\boldsymbol{\theta}_{\mathcal{L}} - \boldsymbol{\alpha}_t)'(\boldsymbol{\theta}_{\mathcal{L}} - \boldsymbol{\alpha}_t) \\
&= [(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} - \boldsymbol{\alpha}_t]'[(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} - \boldsymbol{\alpha}_t] \\
&= [(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'(\mathbf{X}\boldsymbol{\alpha}_t + \mathbf{y}_u) - \boldsymbol{\alpha}_t]'[(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'(\mathbf{X}\boldsymbol{\alpha}_t + \mathbf{y}_u) - \boldsymbol{\alpha}_t] \\
&= [(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}_u]'[(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}_u] \\
R_t(f) &= \mathbf{y}_u'\mathbf{X}(\mathbf{X}'\mathbf{X})^{-2}\mathbf{X}'\mathbf{y}_u
\end{aligned}$$

L'erreur d'estimation est donc une forme quadratique du vecteur $\mathbf{y}_u$. De la même manière, le risque empirique minimal évalué pour $\boldsymbol{\theta} = \boldsymbol{\theta}_{\mathcal{L}}$ s'exprime par une forme quadratique du vecteur $\mathbf{y}_u$[8] :

$$\begin{aligned}
R_{emp}(f) &= \frac{1}{N}(\mathbf{X}\boldsymbol{\theta}_{\mathcal{L}} - \mathbf{y})'(\mathbf{X}\boldsymbol{\theta}_{\mathcal{L}} - \mathbf{y}) \\
&= \frac{1}{N}\mathbf{y}'[\mathbf{I} - \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}']\mathbf{y} \\
&= \frac{1}{N}(\mathbf{y}_t + \mathbf{y}_u)'[\mathbf{I} - \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'](\mathbf{y}_t + \mathbf{y}_u) \\
R_{emp}(f) &= \frac{1}{N}\mathbf{y}_u'[\mathbf{I} - \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}']\mathbf{y}_u
\end{aligned}$$

[8] On utilise les propriétés de la matrice chapeau $\mathbf{H} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$. Le sous-espace vectoriel engendré par les colonnes de $\mathbf{X}$ est stable par $\mathbf{H}$ : $\mathbf{HX} = \mathbf{X}$.

Pour établir un lien entre l'espérance du risque fonctionnel et l'espérance du risque empirique, nous allons procéder par décomposition en valeurs singulières de la matrice des effets $\mathbf{X}$. On supposera que $\mathbf{X}$ est de rang plein (toutes les valeurs singulières sont strictement positives) et que le nombre d'expériences $N$ est supérieur au nombre de régresseurs $p+1$. Soit la décomposition en valeurs singulières de $\mathbf{X}$ :

$$\mathbf{X} = \mathbf{USV}'$$

Compte tenu des propriétés d'orthogonalité sur $\mathbf{U}$ et $\mathbf{V}$, la matrice chapeau $\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$ qui figure dans le risque empirique, et la matrice $\mathbf{X}(\mathbf{X}'\mathbf{X})^{-2}\mathbf{X}'$ qui figure dans l'erreur d'estimation, s'expriment par :

$$\begin{aligned} \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}' &= \mathbf{U}\mathbf{U}' \\ \mathbf{X}(\mathbf{X}'\mathbf{X})^{-2}\mathbf{X}' &= \mathbf{U}\mathbf{S}^{-2}\mathbf{U}' \end{aligned}$$

Nous obtenons donc une écriture plus concise de l'erreur d'estimation et du risque empirique :

$$\begin{aligned} R_t(f) &= \mathbf{y}_u^{'}[\mathbf{U}\mathbf{S}^{-2}\mathbf{U}^{'}]\mathbf{y}_u \\ R_{emp}(f) &= \frac{1}{N}\mathbf{y}_u^{'}[\mathbf{I} - \mathbf{U}\mathbf{U}^{'}]\mathbf{y}_u \end{aligned}$$

Le vecteur $\mathbf{U}^{'}\mathbf{y}_u$ joue un rôle particulier que nous allons analyser. En notant $y_{ui}$ les composantes de $\mathbf{y}_u$, $U_{ik}$ celles des vecteurs colonnes (orthonormés) $\mathbf{U}_i$ de la matrice $\mathbf{U}$, et $\lambda_k$ les $p+1$ valeurs singulières ($k = 0, 1, \ldots, p$), on obtient :

$$\begin{aligned} R_t(f) &= \sum_{k=0}^{p} \frac{1}{\lambda_k^2}[\sum_{i=1}^{N} y_{ui}U_{ik}]^2 \\ R_{emp}(f) &= \frac{1}{N}\sum_{i=1}^{N} y_{ui}^2 - \frac{1}{N}\sum_{k=0}^{p}[\sum_{i=1}^{N} y_{ui}U_{ik}]^2 \end{aligned}$$

Pour une fonction donnée à approcher, lorsque la base des polynômes est fixée, les valeurs $y_{ui}$, $U_{ik}$ dépendent uniquement des points spécifiés par le plan d'expériences. Dans [Chapelle 04], les composantes $y_{ui}$, $U_{ik}$ sont supposées être issues d'un tirage aléatoire et statistiquement indépendantes. Cette indépendance implique la factorisation des espérances.

$$E(y_{ui}U_{ik}) = E(y_{ui})E(U_{ik})$$

Compte tenu de l'orthogonalité des polynômes, l'espérance $E(U_{ik})$ est supposée nulle. Dans [Chapelle 04] ces hypothèses permettent de lier l'espérance du risque fonctionnel à l'espérance du risque empirique en fonction de la trace de la matrice de dispersion.

Nous allons aboutir aux mêmes conclusions que dans [Chapelle 04] en faisant d'autres hypothèses. Nos hypothèses sont basées sur le modèle de la régression. On considère donc les composantes $y_{ui}$, correspondant à la partie tronquée de la solution $\mathbf{y(x)}$, comme les réalisations d'une variable aléatoire d'espérance nulle et de variance $\sigma^2$ :

$$\begin{aligned} E(y_{ui}) &= 0 \\ E(y_{ui}y_{uj}) &= \sigma^2\delta_{ij} \end{aligned}$$

Compte tenu de ces hypothèses, le risque empirique se développe de la façon suivante :

$$\begin{aligned} E[R_{emp}(f)] &= E[\frac{1}{N}\sum_{i=1}^{N} y_{ui}^2 - \frac{1}{N}\sum_{k=0}^{p}[\sum_{i=1}^{N} y_{ui}U_{ik}]^2] \\ &= \frac{1}{N}\sum_{i=1}^{N} E[y_{ui}^2] - \frac{1}{N}\sum_{k=0}^{p} E[[\sum_{i=1}^{N} y_{ui}U_{ik}]^2] \\ &= \sigma^2 - \frac{1}{N}\sum_{k=0}^{p}\sum_{i=1}^{N}\sum_{j=1}^{N} E[y_{ui}y_{uj}]U_{ik}U_{jk} \\ &= \sigma^2 - \frac{1}{N}\sum_{k=0}^{p}\sum_{i=1}^{N} E[y_{ui}^2]U_{ik}^2 \\ &= \sigma^2 - \frac{1}{N}\sum_{k=0}^{p}\sigma^2\sum_{i=1}^{N} U_{ik}^2 \\ &= \sigma^2 - \frac{1}{N}\sum_{k=0}^{p}\sigma^2 \\ E[R_{emp}(f)] &= \sigma^2(1 - \frac{p+1}{N}) \end{aligned}$$

On vérifie que lorsque le nombre $p+1$ de régresseurs est égal au nombre $N$ d'expériences, la régression devient une interpolation : il y a autant de degrés de liberté que d'équations, et le risque empirique s'annule. Le même développement peut être fait sur l'erreur d'estimation $R_t(f)$. En notant $\lambda_k$ les valeurs singulières de $\mathbf{X}$ il vient :

$$\begin{aligned} E[R_t(f)] &= \sum_{k=0}^{p} \frac{1}{\lambda_k^2} E[[\sum_{i=1}^{N} y_{ui} U_{ik}]^2] \\ &= \sum_{k=0}^{p} \frac{\sigma^2}{\lambda_k^2} \sum_{i=1}^{N} E[y_{ui}^2] U_{ik}^2 \\ &= \sum_{k=0}^{p} \frac{\sigma^2}{\lambda_k^2} \sum_{i=1}^{N} U_{ik}^2 \\ E[R_t(f)] &= \sum_{k=0}^{p} \frac{\sigma^2}{\lambda_k^2} \end{aligned}$$

En ajoutant à l'espérance de l'erreur d'estimation, l'espérance de l'erreur de troncature supposée indépendante, on obtient l'espérance du risque fonctionnel :

$$\begin{aligned} E[R(f)] &= E[R_t(f)] + E[y_u^2] \\ &= \sum_{k=0}^{p} \frac{\sigma^2}{\lambda_k^2} + \sigma^2 \\ E[R(f)] &= (1 + \sum_{k=0}^{p} \frac{1}{\lambda_k^2})\sigma^2 \end{aligned}$$

La relation entre l'espérance du risque fonctionnel et celle du risque empirique s'obtient alors en exprimant la variance $\sigma^2$ inconnue en fonction de l'espérance du risque empirique :

$$E[R(f)] = E[R_{emp}(f)](1 - \frac{p+1}{N})^{-1}(1 + \sum_{k=0}^{p} \frac{1}{\lambda_k^2}) \tag{3.5}$$

En notant $\gamma_k$ les valeurs propres de la matrice $\mathbf{X'X}/N$ [9], on retrouve la formule de [Chapelle 04]. Mais rappelons que l'équation 3.5 a été obtenue avec des hypothèses différentes, plus simples et plus conformes au cadre théorique des plans d'expériences.

Posons $d = p + 1$ le nombre de régresseurs et $\gamma_i, i = 1, 2, \ldots, d$ les valeurs propres de la matrice $\mathbf{X'X}/N$ ; on obtient :

$$E[R(f)] = E[R_{emp}(f)](1 - \frac{d}{N})^{-1}(1 + \frac{\sum_{i=1}^{d} 1/\gamma_i}{N}) \tag{3.6}$$

[9] La matrice $\mathbf{X'X}/N$ correspond à la matrice de variance-covariance des vecteurs colonnes de $\mathbf{X}$ en dehors de la première colonne égale à 1 et en supposant évidemment les vecteurs colonnes centrés ce qui est justifié puisque les régresseurs le sont.

L'espérance mathématique du risque fonctionnel est donc le produit de l'espérance mathématique du risque empirique par un facteur $T(\mathcal{L}_N, d)$, qui dépend du nombre $d$ de régresseurs, de la taille $N$ de l'échantillon, et des valeurs propres $\gamma_i$ :

$$T(d, \mathcal{L}_N) = (1 - \frac{d}{N})^{-1}(1 + \frac{\sum_{i=1}^{d} 1/\gamma_i}{N})$$

Lorsque la famille des régresseurs et la taille de l'échantillon sont fixés, la minimisation du facteur multiplicatif $T(d, \mathcal{L}_n)$ revient à déterminer le plan d'expériences de taille $N$ qui minimise la somme des inverses des valeurs propres de la matrice $\mathbf{X'X}/N$. Ce critère correspond à la **A-Optimalité** des plans d'expériences que nous verrons au chapitre 4.

D'autre part, une analyse asymptotique permet d'établir un lien avec le critère d'Akaike [Akaike 70] ou **AIC**[10] Lorsque le nombre de points $N$ devient très grand, la matrice de variance-covariance tend vers la matrice identité, en raison de l'orthogonalité et de la normalisation des régresseurs. Les valeurs propres de la matrice de variance covariance $\mathbf{X'X}/N$ tendent vers 1 :

$$N >> 1 \Rightarrow T(d, \mathcal{L}_N) \quad \simeq \quad (1 - \frac{d}{N})^{-1}(1 + \frac{d}{N})$$

De plus, en supposant $d << N$, ce qui est justifié dans le cas asymptotique, on obtient :

$$E[R(f)] \quad \simeq \quad E[R_{emp}(f)](1 + 2\frac{d}{N})$$

On retrouve donc le rapport $d/N$ figurant dans le critère d'Akaike. Une étude plus approfondie permettrait une analyse plus complète sur la comparaison des 2 critères.

Ce point commun aux approches *plans d'expériences* et *apprentissage statistique* nous est apparu suffisamment intéressant pour être souligné. Nous n'utiliserons pas ce critère sur l'espérance du risque fonctionnel car il necessite d'avoir $E[R_{emp}(f)]$ et qu'il n'est validé que dans le cadre de modèles linéaires par rapport aux paramètres, or, nous utiliserons, dans la suite du mémoire, des modèles non linéaires par rapport aux paramètres comme nous l'avions souligné dans l'introduction générale.

## 3.3 Estimation de l'erreur de généralisation par des techniques de ré-échantillonnage

Nous allons décrire quelques méthodes d'estimation de l'erreur de généralisation par des techniques de ré-échantillonnage. Ces estimations sont nécessaires pour mettre en oeuvre des règles de choix de complexité, et de sélection de modèles.

[10] Acronyme de *Akaike Information Criterion*.

### 3.3.1 Validation croisée

La validation croisée repose sur le principe de n'utiliser qu'une partie des exemples en apprentissage et d'utiliser la partie complémentaire afin d'estimer le risque fonctionnel. La base d'exemples est partitionnée en $D$ parties de cardinalités égales ou équivalentes $\mathcal{L} = \cup\mathcal{L}_i, i = 1, 2, \ldots, D$. Chacun des $D$ apprentissages est réalisé à partir d'une base d'apprentissage tronquée $\mathcal{L}_{(i)} = \mathcal{L} - \mathcal{L}_i$ et l'estimation du risque fonctionnel est réalisée sur la partie complémentaire $\mathcal{L}_i$.

En prenant, par exemple, la norme quadratique et en notant $\texttt{EQMT}_i$ l'erreur quadratique moyenne de test calculée sur la base $\mathcal{L}_i$ par le réseau entraîné sur la base $\mathcal{L}_{(i)}$, le risque fonctionnel quadratique est estimé par la moyenne arithmétique des $D$ erreurs quadratiques moyennes de test $\texttt{EQMT}_i, i = 1, 2, \ldots, D$.

$$R(f) \simeq \frac{1}{D} \sum_{i=1}^{D} \texttt{EQMT}_i$$

---

**Algorithme : Validation Croisée**

- Base $\mathcal{L}$ des exemples ordonnés `aléatoirement`
- Réaliser une partition de $\mathcal{L} = \cup_{i=1}^{D}\mathcal{L}_i, \mathcal{L}_i \cap \mathcal{L}_j = \emptyset, j \neq i$
- Pour $i = 1, 2, \ldots, D$ faire :
  - Apprentissage du modèle sur la base $\mathcal{L}_{(i)} = \mathcal{L} - \mathcal{L}_i$
  - Calculer l'erreur moyenne de test $\texttt{EQMT}_i$ sur la base $\mathcal{L}_i$
- Estimer le risque fonctionnel par $R(f) \simeq (1/D) \sum_{i=1}^{D} \texttt{EQMT}_i$

---

Cet algorithme nécessite le choix du nombre de découpages. Le découpage de la base d'exemples dépend évidemment de l'ordre initial des exemples. Lorsque les exemples sont spécifiés par une méthode déterministe, il faut au préalable mélanger aléatoirement les exemples. En effet, dans le cas par exemple d'un maillage régulier, un découpage sans mélange préalable des exemples produit des bases $\mathcal{L}_{(i)}$ peu représentatives de l'ensemble de l'échantillon.

### 3.3.2 Le Leave-One-Out

Comme son nom l'indique, le *Leave-One-Out* est une méthode qui teste le modèle sur l'exemple qui n'a pas été utilisé en apprentissage. La méthode est similaire à la validation croisée avec $\mathcal{L}_i = \{(\mathbf{x}_i, y_i)\}$, mais la partition est déterministe.

**Algorithme : Leave-One-Out**

- Base d'exemples $\mathcal{L} = \{(\mathbf{x}_i, y_i), i = 1, 2, \ldots, N\}$
- Pour $i = 1, 2, \ldots, N$ faire :
  - Apprentissage du modèle sur la base $\mathcal{L}_{(i)} = \mathcal{L} - \{(\mathbf{x}_i, y_i)\}$
  - Calculer l'erreur de test $\texttt{EQMT}_i$ sur l'exemple $\{(\mathbf{x}_i, y_i)\}$
- Estimer le risque fonctionnel par $R(f) \simeq (1/N) \sum_{i=1}^{N} \texttt{EQMT}_i$

Cette méthode reste évidemment limitée aux problèmes pour lesquels le nombre d'exemples est petit dans la mesure où on devra réaliser autant d'apprentissages qu'il y a d'exemples dans la base d'apprentissage. Ce problème ne se pose pas dans le cas où les fonctions de $\mathcal{F}$ sont linéaires par rapport aux paramètres $\boldsymbol{\theta}$ dans la mesure où l'on peut utiliser la propriété du LOO Virtuel vue au chapitre 2 lors de la présentation des leviers. Pour des modèles non linéaires par rapport aux paramètres $\boldsymbol{\theta}$, le LOO virtuel peut également être utilisée en approchant la fonction paramétrique par un développement au premier ordre, et en se ramenant à un problème linéaire par rapport aux paramètres comme dans [Monari & Dreyfus 00].

Nous allons présenter l'estimation de l'erreur de généralisation par LOO virtuel pour les modèles linéaires puis pour les modèles non linéaires.

### le LOO virtuel pour les modèles linéaires

Rappelons que nous notons, $\mathbf{X}$ la matrice des effets, $h_{ii}$ les termes diagonaux de la matrice $\mathbf{H} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$ et $r_i = y_i - f(\mathbf{x}_i, \boldsymbol{\theta}_{\mathcal{L}})$ les résidus ; alors l'estimation de l'erreur de généralisation s'obtient par :

$$R(f) \simeq \frac{1}{N} \sum_{i=1}^{N} (\frac{r_i}{1 - h_{ii}})^2 \tag{3.7}$$

On retrouve, dans l'expression de l'erreur de généralisation, le rôle joué par les leviers.

### LOO virtuel pour les modèles non linéaires

On fait un développement au premier ordre de $f(\mathbf{x}, \boldsymbol{\theta})$ autour de $\boldsymbol{\theta}_0$

$$f(\mathbf{x}, \boldsymbol{\theta}) = f(\mathbf{x}, \boldsymbol{\theta}_0) + \mathbf{Z}(\boldsymbol{\theta} - \boldsymbol{\theta}_0) \tag{3.8}$$

où $\mathbf{Z}$ est la matrice jacobienne du modèle. Elle contient $N$ lignes et $p$ colonnes et est définie comme :

$$[\mathbf{Z}] = [\frac{\partial f(\mathbf{x}_i, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}]_{\boldsymbol{\theta}=\boldsymbol{\theta}_0}$$

on a donc un modèle linéaire en $\boldsymbol{\theta}$.

On applique la méthode classique des moindres carrés en notant $\mathbf{y}$ le vecteur des composantes $y_i$, on a alors :

$$\begin{aligned}\boldsymbol{\theta}_{mc} &= (\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'(\mathbf{y} - f(\mathbf{x}, \boldsymbol{\theta}_0) + \mathbf{Z}\boldsymbol{\theta}_0) \\ \boldsymbol{\theta}_{mc} &= \boldsymbol{\theta}_0 + (\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'(\mathbf{y} - f(\mathbf{x}, \boldsymbol{\theta}_0))\end{aligned}$$

Ce développement peut être fait autour de la valeur $\boldsymbol{\theta}_{\mathcal{L}}$ correspondant à la solution optimale des moindres carrés calculée sur la totalité des exemples :

$$\boldsymbol{\theta}_{mc} = \boldsymbol{\theta}_{\mathcal{L}} + (\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'(\mathbf{y} - f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}}))$$

Avec les hypothèses précédentes, on peut alors appliquer la règle du LOO virtuel comme pour tout modèle linéaire par rapport aux paramètres. En rappelant que $\mathbf{Z}$ est la matrice jacobienne évaluée en $\boldsymbol{\theta}_{\mathcal{L}}$, $h_{ii}$ les termes diagonaux de la matrice $\mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}^{'}$ et $r_i$ le résidu de l'exemple $i$, l'estimation de l'erreur de généralisation par leave one out virtuel s'obtient par :

$$\begin{aligned}\boldsymbol{\theta}_{\mathcal{L}} &= \texttt{ArgMin}_{\boldsymbol{\theta}} \sum_{i=1}^{N} (f(\mathbf{x}_i, \boldsymbol{\theta}) - y_i)^2 \\ \mathbf{Z} &= [\frac{\partial f(\mathbf{x}_i, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}]_{\boldsymbol{\theta}=\boldsymbol{\theta}_{\mathcal{L}}} \\ h_{ii} &= [\mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}']_{ii}\end{aligned}$$

$$R(f) \simeq \frac{1}{N}\sum_{i=1}^{N}(\frac{f(\mathbf{x}_i, \boldsymbol{\theta}_{\mathcal{L}}) - y_i}{1 - h_{ii}})^2 \tag{3.9}$$

Rappelons que cette estimation est fondée sur l'hypothèse que le retrait d'un exemple a sur les paramètres du modèle une influence suffisamment petite pour que le développement limité au premier ordre (equation (3.8)) soit justifié.

### 3.3.3 Estimation de l'erreur de généralisation par Bootstrap

Le principe de l'estimation de l'erreur de généralisation par bootstrap est de créer $B$ répliques de la base d'apprentissage. Notons par $\mathcal{L}^{*b}$ la réplique $b$ de la base d'apprentissage initiale $\mathcal{L}$.

Nous munirons, en général, de l'indice supérieur $*b$ les quantités relatives à la réplique $b$.

Rappelons qu'en moyenne, 37% des exemples de la base d'apprentissage n'apparaissent pas dans une réplique. Comme décrit dans la figure 2.7, on engendre un modèle par réplique. Chacune des répliques constitue donc une base d'apprentissage différente, et la validation du modèle qui a appris sur cette base est effectuée sur les éléments de la base initiale.

L'erreur d'apprentissage calculée sur la réplique et l'erreur de validation calculée sur la base initiale sont considérées comme des réalisations de variables aléatoires représentatives de l'erreur d'apprentissage et de l'erreur de généralisation. Nous pouvons, pour chacune des répliques, calculer l'erreur quadratique moyenne d'apprentissage $E_a^{*b}$ et l'erreur quadratique moyenne de validation $E_v^{*b}$ de la manière suivante :

$$\begin{aligned} E_a^{*b} &= \frac{1}{N}\sum_{k=1}^{N}(y_k^{*b} - f(\mathbf{x}_k^{*b}, \boldsymbol{\theta}_{\mathcal{L}^{*b}}))^2 \\ E_v^{*b} &= \frac{1}{N}\sum_{k=1}^{N}(y_k - f(\mathbf{x}_k, \boldsymbol{\theta}_{\mathcal{L}^{*b}}))^2 \end{aligned}$$

où :

- $E_a^{*b}$ et $E_v^{*b}$ désignent l'erreur d'apprentissage et l'erreur de validation de la réplique $b$,
- $y_k^{*b}$ désigne la valeur de la réponse du processus à modéliser pour le $k^{\texttt{ième}}$ exemple de la réplique $b$ ; $y_k^{*b}$ est généralement différent de $y_k$, qui est la valeur de la réponse du processus à modéliser pour le $k^{\texttt{ième}}$ exemple de la base d'apprentissage initiale,
- $\mathbf{x}_k^{*b}$ désigne le vecteur des facteurs du $k^{\texttt{ième}}$ exemple de la réplique $b$, qui est généralement différent de $\mathbf{x}_k$ qui est le vecteur des facteurs du $k^{\texttt{ième}}$ exemple de la base d'apprentissage initiale,
- $\boldsymbol{\theta}_{\mathcal{L}^{*b}}$ désigne le vecteur des paramètres du modèle créé à partir de la réplique $b$ notée $\mathcal{L}^{*b}$,
- $f(\mathbf{x}_k^{*b}, \boldsymbol{\theta}_{\mathcal{L}^{*b}})$ désigne la réponse prédite par le modèle créé à partir de la réplique $b$, lorsqu'on lui présente le $k^{\texttt{ième}}$ exemple de celle-ci,
- $f(\mathbf{x_k}, \boldsymbol{\theta}_{\mathcal{L}^{*b}})$ désigne la réponse du modèle créé à partir de la réplique $b$ lorsqu'on lui présente le $k^{\texttt{ième}}$ exemple de la base d'apprentissage initiale.

Après avoir obtenu $B$ réalisations de ces erreurs, nous pouvons établir la moyenne et la variance de $E_a$ et de $E_v$. $E_a$ et $E_v$ sont les variables aléatoires dont les réalisations pour la réplique $b$ sont $E_a^{*b}$ et $E_v^{*b}$. Notons par $m_{\mathcal{B}}(E_a)$, $m_{\mathcal{B}}(E_v)$, $s^2_{\mathcal{B}}(E_a)$ et $s^2_{\mathcal{B}}(E_v)$ respectivement les moyennes et variances de $E_a$ et $E_v$.

$$
\begin{aligned}
m_{\mathcal{B}}(E_a) &= \frac{1}{B}\sum_{b=1}^{B} E_a^{*b} \\
s^2_{\mathcal{B}}(E_a) &= \frac{1}{B}\sum_{b=1}^{B} (E_a^{*b} - m_{\mathcal{B}}(E_a))^2
\end{aligned}
$$

et

$$
\begin{aligned}
m_{\mathcal{B}}(E_v) &= \frac{1}{B}\sum_{b=1}^{B} E_v^{*b} \\
s^2_{\mathcal{B}}(E_v) &= \frac{1}{B}\sum_{b=1}^{B} (E_v^{*b} - m_{\mathcal{B}}(E_v))^2
\end{aligned}
$$

Nous verrons, dans le paragraphe concernant le choix de la complexité, que la moyenne de l'erreur de validation constitue une très bonne approximation de l'erreur de généralisation.

Nous avions souligné qu'un inconvénient de la validation croisée était le choix arbitraire de la partition en $D$ sous-ensembles de la base d'apprentissage. Une ou plusieurs partitions peuvent ne pas être représentatives de la distribution des exemples de la base d'apprentissage complète, ce qui peut engendrer une mauvaise estimation de l'erreur. Il se peut également qu'une réplique ne soit pas représentative de la base d'apprentissage, surtout lorsque le cardinal de cette base est faible. Cependant, il faut souligner qu'en pratique, nous faisons 200 répliques bootstrap, ce qui confère une meilleur stabilité à la méthode dans le sens où les bases non représentatives n'ont pas une grande influence sur la statistique établie lorsque leur nombre est faible. De plus, on peut utiliser des statistiques bootstrap n'utilisant que des quantiles de la distribution des erreurs des $B$ modèles. On peut prendre, par exemple, 90% des meilleurs modèles pour établir les statistiques décrites précédemment. En effet, du fait des minima locaux de la fonction de coût, lorsque le modèle n'est pas linéaire, certains des $B$ modèles peuvent avoir de mauvaises performances de généralisation. En ne prenant qu'une partie des $B$ modèles, la statistique est plus précise puisque les modèles ayant de mauvaises performances, dues à une réplique non représentative ou à un jeu de paramètres du modèle obtenu à partir d'un minimum local de la fonction de coût, sont alors filtrés.

## 3.4 Choix d'une complexité

### 3.4.1 Principe

Nous venons de décrire des méthodes qui permettent d'estimer l'erreur de généralisation. Celles-ci sont mises en oeuvre au sein de procédures qui consistent à observer l'évolution de l'erreur de généralisation en fonction de la complexité du modèle. La complexité optimale est celle que l'on obtient, en augmentant pas à pas la complexité du modèle, avant d'observer une augmentation de l'erreur de généralisation due au surajustement du modèle. Dans le cas des modèles non linéaires, il faut prendre soin, pour chaque complexité, de créer plusieurs modèles

obtenus avec des initialisations différentes des paramètres au début de l'apprentissage, et de prendre, par exemple, la plus petite erreur de généralisation parmi celles de l'ensemble des modèles générés.

Les modèles que nous utilisons dans ce mémoire sont les polynômes, pour le cadre linéaire, et les réseaux de neurones, pour le cadre non linéaire. La complexité se traduit donc en termes de degré du polynôme pour les premiers et de nombre de neurones cachés pour les seconds. Nous allons présenter une méthode de sélection de la complexité utilisant l'estimation de l'erreur de généralisation par bootstrap que nous avons décrite précédemment. Nous allons l'appliquer à des modèles polynomiaux pour nous affranchir des problèmes de minima locaux, mais cette méthode pourra tout à fait être étendue à la sélection de modèles non linéaires en leurs paramètres.

Notons par $f_c(\mathbf{x}, \boldsymbol{\theta})$ le modèle de complexité $c$. L'objectif est de déterminer, uniquement avec la base d'apprentissage de $N$ points dont nous disposons, la complexité $c$ qui permet de minimiser au mieux, sur l'ensemble du domaine d'étude $\mathcal{D}$, l'erreur de généralisation notée $R(f_c)$ :

$$R(f_c) = \int_{\mathcal{D}} (y(\mathbf{x}) - f_c(\mathbf{x}, \boldsymbol{\theta}))^2 P(\mathbf{x}) d\mathbf{x}$$

où $y(\mathbf{x})$ est la sortie du processus à modéliser pour le vecteur d'entrée $\mathbf{x}$.

Pour déterminer cette complexité $c$ optimale par bootstrap, nous faisons varier le degré du polynôme et nous estimons l'erreur de généralisation par bootstrap pour chacun d'eux comme nous l'avons décrit au paragraphe 3.3.3 à l'aide de la moyenne de l'erreur de validation. Notons par $R_{\mathcal{B}}(f_c)$ cette erreur estimée par bootstrap.

$$R_{\mathcal{B}}(f_c) = \frac{1}{B} \sum_{b=1}^{B} \big[ \frac{1}{N} \sum_{k=1}^{N} (y_k - f_c(\mathbf{x}_k, \boldsymbol{\theta}_{\mathcal{L}^{*b}}))^2 \big]$$

Afin de valider la méthode, nous allons comparer l'évolution de l'erreur de généralisation obtenu par bootstrap $R_{\mathcal{B}}(f_c)$ avec une estimation de l'erreur de généralisation obtenue à l'aide d'un maillage fin du domaine d'étude $\mathcal{D}$. Nous noterons par $R_m(f_c)$ cette erreur.

$$R_m(f_c) = \frac{1}{N_m} \sum_{i=1}^{N_m} (y_i - f_c(\mathbf{x}_i, \boldsymbol{\theta}))^2 \tag{3.10}$$

où $N_m$ est le nombre d'exemples du maillage et $y_i$ la sortie associée au point de maillage $\mathbf{x}_i$. Nous prendrons un maillage contenant plusieurs milliers d'exemples : $R_m(f_c)$ sera alors une très bonne approximation de l'erreur de généralisation $R(f_c)$.

Enfin, nous utiliserons également une estimation, par validation croisée, de l'erreur de généralisation pour la complexité $c$. L'estimation sera calculée de la manière suivante :

$$R_{VC}(f_c) = \frac{1}{D} \sum_{i=1}^{D} \big[ \frac{1}{N_V} \sum_{k=1}^{N_V} (y_k - f_c(\mathbf{x}_k, \boldsymbol{\theta}_{\mathcal{L}_{(i)}}))^2 \big]$$

où $N_V = \frac{N}{D}$ représente le nombre d'exemples du sous-ensemble qui sert de base de validation ; nous utiliserons la validation croisée avec $D = 5$. Nous avons noté par $\boldsymbol{\theta}_{\mathcal{L}_{(i)}}$ le jeu de paramètres obtenu à partir de la base d'apprentissage sans le ième sous-ensemble de la partition utilisée.

Nous vérifierons si les minima de $R_{\mathcal{B}}(f_c)$, de $R_{VC}(f_c)$ et de $R_m(f_c)$ sont atteints pour des complexités équivalentes.

### 3.4.2 Quelques exemples

Nous allons utiliser, dans ce paragraphe, quelques exemples simples pour mettre en évidence l'évolution de l'erreur de généralisation estimée par bootstrap et par validation croisée en fonction de la complexité $c$ du polynôme. Nous montrons également l'erreur de généralisation approchée par $R_m(f_c)$. Nous verrons si ces deux méthodes permettent de déterminer la complexité optimale des apprentis pour approcher chacune des fonctions utilisées.

#### Problèmes liés aux méthodes de ré-échantillonnage

Rappelons les précautions à prendre lorsqu'on fait appel aux méthodes de ré-échantillonnage : le nombre d'exemples distincts doit être supérieur au nombre de paramètres du modèle apprenti. Pour la validation croisée, le problème est simple : il suffit d'assurer que le nombre d'exemples utilisés en apprentissage reste supérieur au nombre de paramètres. Pour le bootstrap, on ne peut pas garantir cette condition. Néanmoins, comme nous l'avons déjà montré au paragraphe 2.4.3, une analyse probabiliste peut nous garantir cette condition avec un niveau de confiance suffisant. La figure 3.1 (déjà présentée au paragraphe 2.4.3) montre la probabilité cumulée de l'occurrence d'une réplique en fonction de son taux d'exemples distincts.

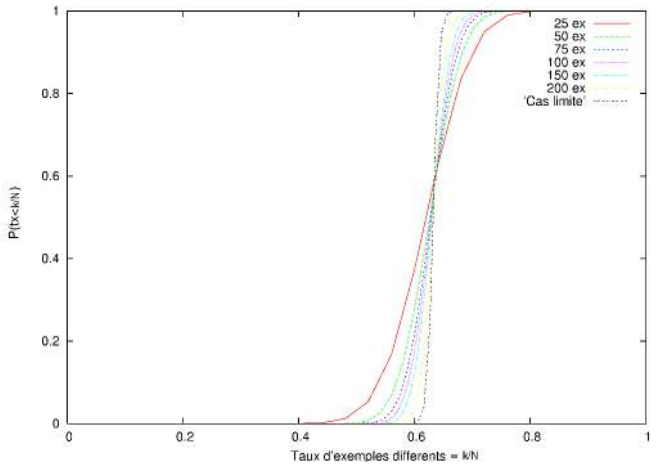



FIG. 3.1 – Pourcentage de répliques pour lesquelles le taux d'exemples différents est inférieur à $X$

Ainsi, pour $N = 50$, le pourcentage de répliques pour lequel le taux d'exemples différents est inférieur à 0.5 est quasiment nul. Autrement dit, l'ensemble des répliques sera constitué d'au moins 25 exemples différents. L'estimation de l'erreur de généralisation obtenue à partir de ces répliques permet d'estimer la complexité optimale d'un modèle contenant au plus 25 paramètres.

Au-delà, il existe des répliques pour lesquelles le nombre de paramètres est supérieur au nombre d'exemples.

**Résultats sur des modèles polynomiaux**

Nous allons utiliser différentes fonctions de $R$ dans $R$ approchée par des polynômes de degré $c$, et allons comparer le degré optimal obtenu par minimisation de l'estimation de l'erreur de généralisation $R_m(f_c)$ à ceux obtenus par bootstrap et validation croisée. L'estimation de l'erreur de généralisation $R_m(f_c)$ est calculée par la moyenne empirique des écarts quadratiques (équation 3.10) évalués sur un maillage régulier de 2000 points.

**La fonction sinus cardinal sur le domaine $\mathcal{D} = [-4;10]$ :**

La figure 3.2 présente les points de la base d'apprentissage utilisés pour estimer la complexité optimale du polynôme pour ce problème.

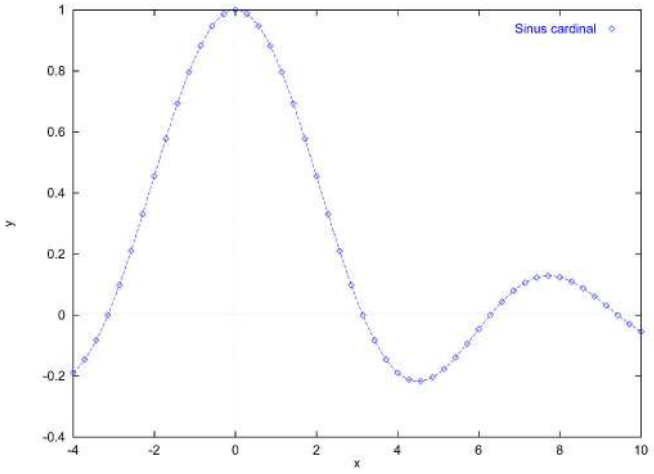



FIG. 3.2 – Base d'exemples de la fonction sinus cardinal

La figure 3.3 montre l'évolution des fonctions $R_{\mathcal{B}}(f_c)$, $R_{VC}(f_c)$ et $R_m(f_c)$ qui représentent respectivement l'erreur de généralisation pour la complexité $c$ estimée par bootstrap, par validation croisée et celle obtenue à l'aide d'un maillage.

Nous pouvons remarquer qu'avec l'estimation par Bootstrap, nous trouvons la véritable complexité optimale ($c = 14$) uniquement avec les 50 exemples de la base d'apprentissage, alors que la valeur théorique a été obtenue avec 2000 exemples. Par la validation croisée, nous trouvons $c = 11$.

Les grandes valeurs de l'erreur de généralisation pour des degrés de polynômes élevés sont dues au sur-ajustement des modèles (voir figure 3.4).

**La fonction cosinus sur l'intervalle $\mathcal{D} = [-\pi;\pi]$ :**

Nous avons conduit la même étude sur la fonction cosinus dans l'intervalle $[-\pi;\pi]$ dont voici les résultats :

La complexité optimale est $c = 12$. Nous trouvons, figure 3.5, le degré 12 par bootstrap et le degré 11 par validation croisée.

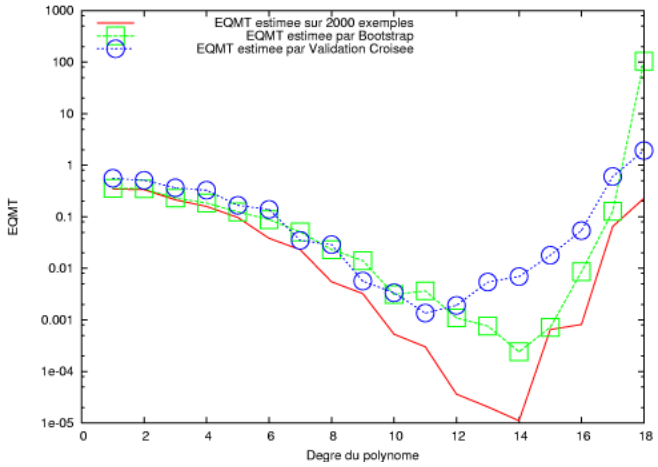



FIG. 3.3 – Erreurs de généralisation (EQMT) estimées par bootstrap, par validation croisée, et à partir d'un maillage de 2000 points en fonction de la complexité de la famille de fonctions ; il s'agit ici de polynômes : l'axe des abscisses représente donc le degré du polynome.

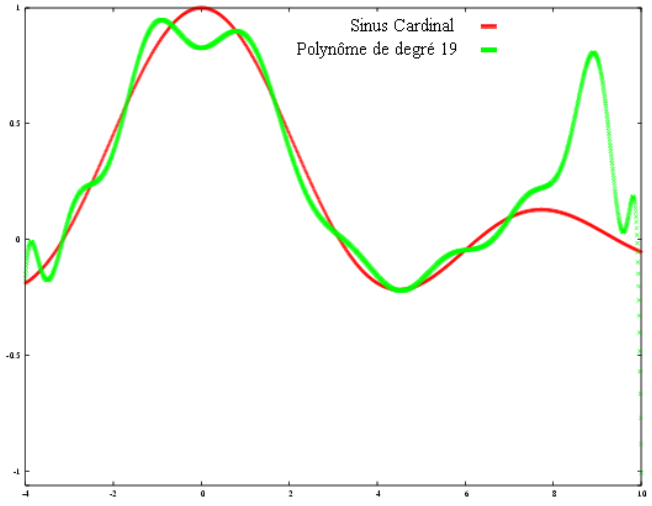



FIG. 3.4 – Réponse du modèle polynomial de degré 19 construit à partir de la base LHS du sinus cardinal.

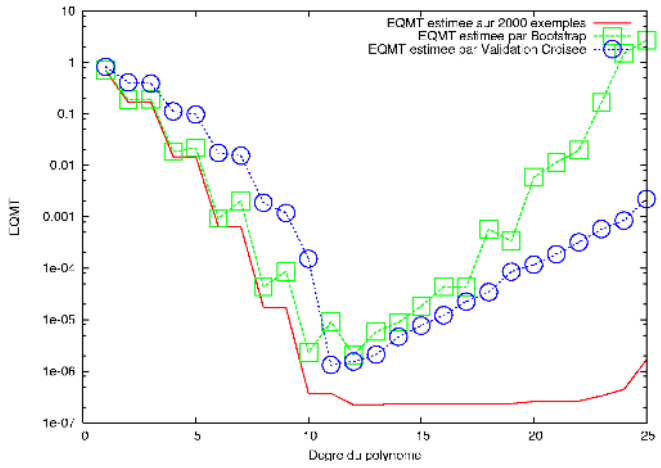



FIG. 3.5 – Erreurs de généralisation estimées par bootstrap, par validation croisée et à partir d'un maillage de 2000 points en fonction du degré du polynôme.

**La fonction $sin(x)cos(0.25x)$ sur l'intervalle $\mathcal{D} = [-4; 10]$ :**

La figure 3.6, représente la fonction $sin(x)cos(0.25x)$ sur l'intervalle $\mathcal{D}$.

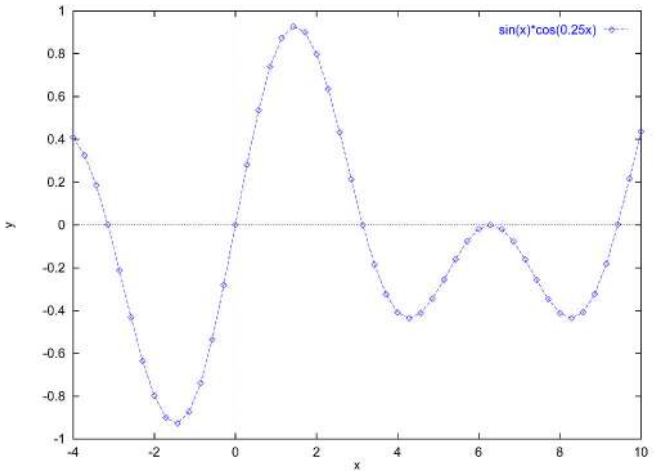



FIG. 3.6 – La fonction $sin(x)cos(0.25x)$ sur l'intervalle $[-4; 10]$

La figure 3.7, présente les résultats obtenus pour cette fonction. Le degré otimal $c = 13$ a été estimé par bootstrap. La validation croisée fournit un polynôme de degré $c = 11$ donc un modèle apprenti sous dimensionné. Notons, néanmoins, qu'il existe une variabilité relativement importante de l'estimation de l'erreur de généralisation par bootstrap. Cela peut s'expliquer par le faible nombre d'exemples pour représenter une fonction assez oscillante.

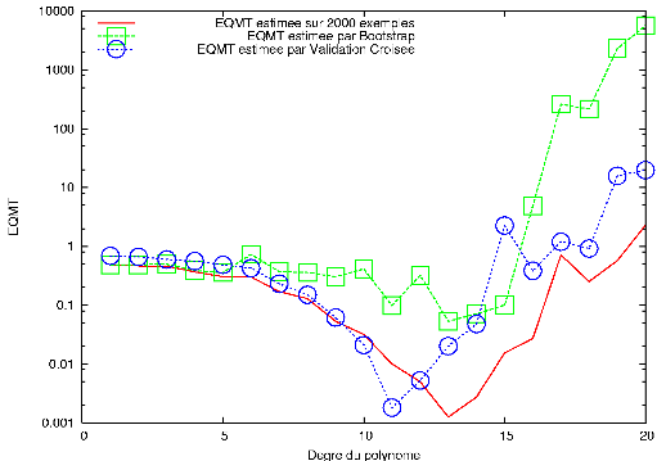



FIG. 3.7 – Erreurs de généralisation estimées par bootstrap, par validation croisée et à partir d'un maillage de 2000 points en fonction du degré du polynôme.

**La fonction $x^{sin(x)}$ sur l'intervalle $\mathcal{D} = [0; 16]$ :**

La figure 3.8 représente la fonction $x^{sin(x)}$ dans l'intervalle considéré. Les résultats de l'estimation de la complexité optimale sont présentés à la figure 3.9.

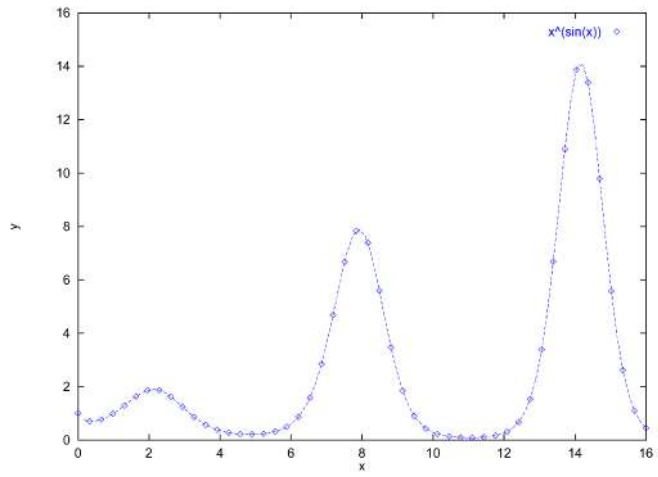



FIG. 3.8 – La fonction $x^{sin(x)}$ sur l'intervalle $[0; 16]$

La famille des polynômes ne permet pas d'approcher convenablement la fonction $x^{sin(x)}$ avec le nombre de points dont on dispose. Il faudrait donc utiliser une autre famille de fonctions, comme les réseaux de neurones par exemple. Cependant, il existe tout de même une architecture optimale correspondant à $c = 11$. Nous trouvons $c = 7$ par bootstrap et $c = 10$ par validation croisée. Cependant, nous pouvons voir que pour ce cas, l'erreur de généralisation varie peu entre les complexités 7 et 13.

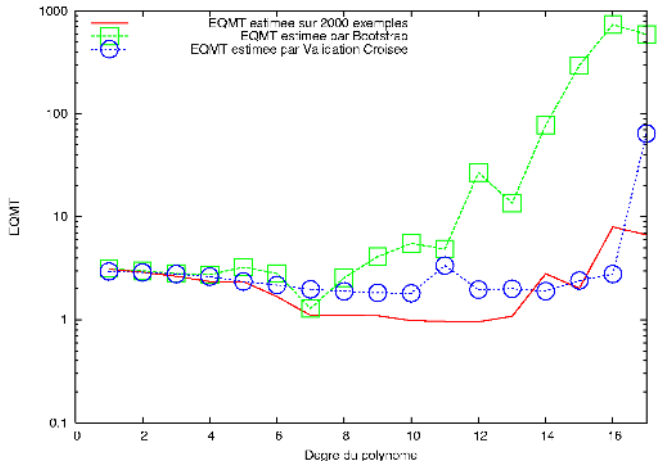



FIG. 3.9 – Erreurs de généralisation estimées par bootstrap et par validation croisée et erreur de généralisation obtenue à partir d'un maillage de 2000 points en fonction du degré du polynome.

Ces exemples montrent que malgré une estimation peu fiable de l'erreur, la validation croisée et le bootstrap permettent de choisir une complexité adaptée des modèles qui doivent être construits. Il faut souligner qu'il est possible de déterminer cette complexité optimale $c$ uniquement avec les points de la base d'apprentissage constituée d'un faible nombre d'exemples par rapport à celui du maillage.

## 3.5 Heuristiques pour l'initialisation des paramètres et pour la régularisation par arrêt prématuré

Cette section ne traite que des modèles non linéaires par rapport aux paramètres, car l'arrêt prématuré, d'une part, et la problématique de choix de modèle pour une complexité donnée, d'autre part, ne se posent pas pour un modèle linéaire par rapport aux paramètres. En effet, dans ce dernier cas, la fonction de coût ne présente qu'un seul minimum. Comme nous l'avons indiqué précédemment, il faut faire des apprentissages de modèles non linéaires en leurs paramètres avec différentes initialisations de ceux-ci. On peut alors obtenir plusieurs modèles de même complexité, de paramètres différents. Il faut donc effectuer un choix entre ces modèles.

L'erreur de généralisation peut être estimée par validation croisée, par leave-one-out virtuel ou par bootstrap. Nous allons étudier une méthode fondée sur le score de leave-one-out virtuel ; nous examinerons ensuite le choix de modèles de complexité fixée dans le cadre des processus déterministes. Enfin, nous étudierons la régularisation de l'apprentissage par arrêt prématuré en estimant l'erreur de généralisation par Bootstrap.

### 3.5.1 Choix du meilleur modèle par leave-one-out virtuel pour une architecture donnée

Dans le cadre du leave-one-out, le choix porte sur le modèle ayant le plus petit score de leave-one-out virtuel (équation (3.9)). Cependant, plusieurs modèles peuvent avoir des scores de

leave-one-out virtuel équivalents, ce qui leurs confère des capacités de généralisation identiques[11]. Lorsque nous avons décrit les leviers, nous avons souligné qu'il était préférable d'avoir un modèle qui soit influencé de manière identique par tous les exemples, ce qui évite le surajustement. Dans ce cas, tous les leviers sont égaux à $\frac{p}{N}$. Dans [Monari 99] et [Monari & Dreyfus 02], les auteurs proposent de caractériser la distribution des leviers par la quantité $\mu$ définie par :

$$\mu = \frac{1}{N} \sum_{k=1}^{N} \sqrt{\frac{N}{p} h_{kk}} \tag{3.11}$$

Cette quantité est inférieure à 1 ; elle vaut 1 si tous les leviers sont égaux à $\frac{p}{N}$ : plus $\mu$ est proche de 1, plus la distribution des leviers est étroite autour de $\frac{p}{N}$. Pour des modèles ayant des capacités de généralisation du même ordre de grandeur, il est préférable de choisir le modèle pour lequel $\mu$ est le plus proche de 1.

Rappelons que ce paragraphe ne concerne que les modèles non linéaires par rapport à leurs paramètres. Dans ce cas, la matrice des effets $\mathbf{X}$ est remplacée par la matrice jacobienne du modèle $\mathbf{Z}$ pour le calcul des leviers. Dans ce cas, les leviers $h_{ii}$ sont les éléments diagonaux de la matrice $\mathbf{H} = \mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'$.

### 3.5.2 Cas particulier des processus déterministes

Les modèles construit à partir d'un processus déterministe souffrent moins de surajustement que ceux construits à partir de données bruitées. En effet, nous avons souligné, à plusieurs reprises, que des modèles sur dimensionnés ou des modèles ayant subit un trop grand nombre de cycles d'apprentissage peuvent s'ajuster au bruit. Dans le cadre déterministe, les modèles ne peuvent pas avoir ce type de comportement car le bruit est inexistant. Le surajustement ne dépend alors que de la complexité du modèle choisi.

Pour illustrer cela, nous avons réalisé, pour une architecture donnée, 500 apprentissages initialisés de manière différente sur la fonction sinus cardinal en dimension 2. Nous avons visualisé, l'erreur quadratique moyenne de test (EQMT) calculée sur un maillage contenant 10000 points en fonction de l'erreur quadratique moyenne d'apprentissage (EQMA) calculée sur la base d'apprentissage contenant 200 points.

Nous pouvons voir à la figure 3.10 que, dans le cadre déterministe, l'EQMT et l'EQMA sont fortement corrélées. Pour la même complexité, les meilleurs modèles en apprentissage sont également les meilleurs modèles en généralisation.

Nous avons reconduit la même étude sur une fonction de $\mathcal{R}^3$ dans $\mathcal{R}$ du processus de Homma et Saltelli [Saltelli & Homma 92] que nous verrons en détail au dernier chapitre. Il est inutile d'expliciter pour l'instant cette fonction mais nous verrons que c'est une fonction difficile à modéliser. Nous avons visualisé également l'EQMT calculée sur un maillage de 20000 points en fonction de l'EQMA obtenue sur la base d'apprentissage contenant 100 points. La figure 3.11 présente les résultats.

La corrélation est moins forte dans cas. Il faut souligner que nous n'avons que 100 exemples pour un problème dont l'espace des entrées est de dimension 3. Nous avons donc reconduit

[11] Ce cas de figure peut également se présenter pour des modèles d'architecture différente. Un modèle moins complexe peut généraliser aussi bien qu'un modèle plus complexe. La méthode que nous décrivons dans ce paragraphe peut être utilisée également dans ce cas de figure.

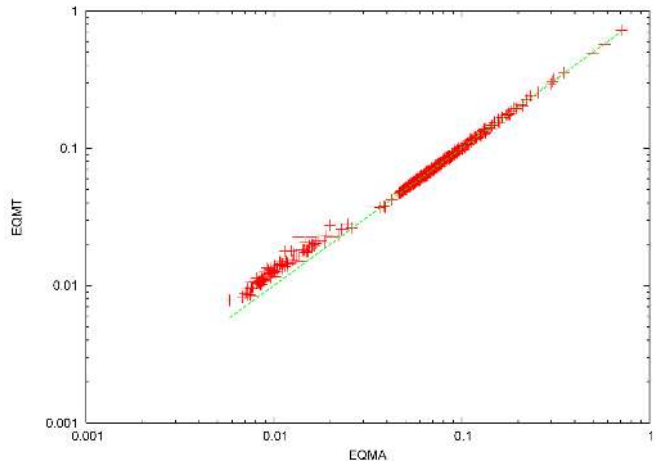



FIG. 3.10 – Corrélations entre l'erreur quadratique moyenne d'apprentissage et l'erreur quadratique moyenne de test dans le cadre d'un processus déterministe et pour des réseaux de neurones de même complexité.

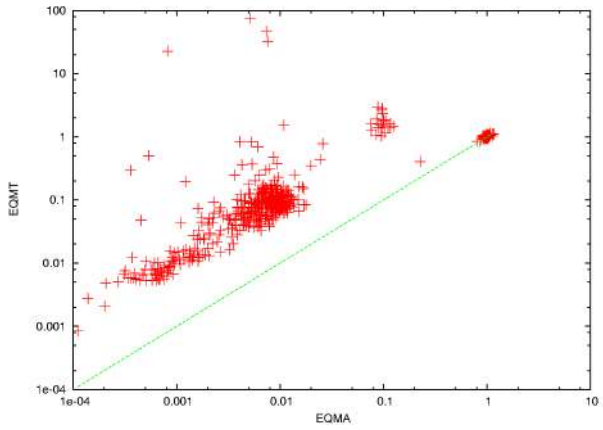



FIG. 3.11 – Corrélation entre l'erreur quadratique moyenne d'apprentissage et l'erreur quadratique moyenne de test dans le cadre du processus déterministe de Homma et Saltelli pour une base comportant 100 exemples et pour des réseaux de neurones de même complexité.

l'expérience avec la même fonction à partir d'un base d'apprentissage de 200 exemples (figure 3.12).

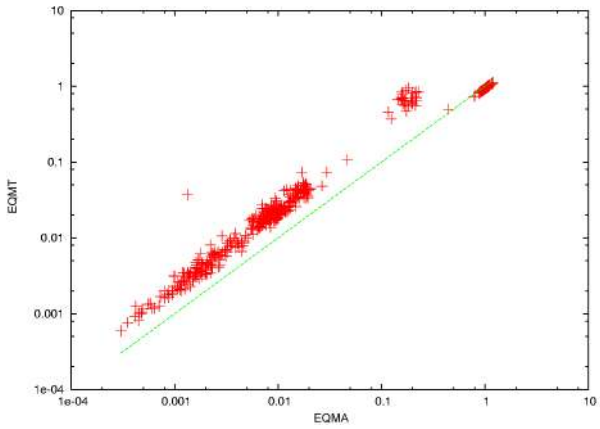



FIG. 3.12 – Corrélations entre l'erreur quadratique moyenne d'apprentissage et l'erreur quadratique moyenne de test dans le cadre du processus déterministe de Homma et Saltelli pour une base comportant 200 exemples et pour des réseaux de neurones de même complexité.

Cette étude nous a permis de modifier la procédure bootstrap que nous utilisons en régression. Nous avons précisé au paragraphe 3.3.3 que l'on pouvait prendre 90 % des meilleurs modèles bootstrap pour filtrer les modèles les moins efficaces qui auraient été obtenus à partir de minima locaux de la fonction de coût. En prenant en considération les remarques concernant les résultats de la figure 3.10, nous mettons en place une stratégie de choix de l'initialisation des paramètres qui sera appliquée à l'ensemble des modèles dont la base d'apprentissage est une réplique bootstrap. Pour déterminer cette initialisation, on entraine un grand nombre de réseaux en sauvegardant le vecteur des paramètres initiaux. Le vecteur de paramètres qui a permis d'obtenir le modèle avec l'EQMA la plus faible sera retenu pour initialiser l'ensemble des modèles dont la base d'apprentissage est une réplique bootstrap. La figure 3.13 présente la démarche.

Ce choix est confirmé par l'observation de la corrélation entre l'EQMA et l'EQMT pour des processus déterministes pour une complexité fixée. La variance bootstrap des sorties estimées n'est pas sensible, dans ce cas, à l'initialisation, car le jeu de paramètres initial est le même pour tous les modèles. La variance bootstrap traduit alors la sensibilité de l'aprentissage par rapport au tirage du ré-échantillonnage bootstrap.

La figure 3.14 résume la démarche à l'aide d'un schéma conceptuel. Cette précaution rappelle les travaux de John Moody [MOODY 94] sur la validation croisée non linéaire. Il propose d'entrainer des modèles neuronaux sur des bases de validation croisée en les initialisant de manière identique à partir d'un jeu de paramètres $\boldsymbol{\theta}_0$.

### 3.5.3 Procédure utilisée pour la construction des modèles par bootstrap

Nous pouvons utiliser le bootstrap pour le choix d'une complexité comme nous l'avons décrit en début de chapitre. Nous avons examiné des exemples de modèles linéaires en leurs paramètres, et nous utiliserons la méthode que nous avons décrite pour déterminer l'architecture optimale des réseaux de neurones que nous utliserons sur les benchmarks du dernier chapitre. Ce problème de choix de complexité est commun aux modèles linéaires et aux modèles non linéaires.

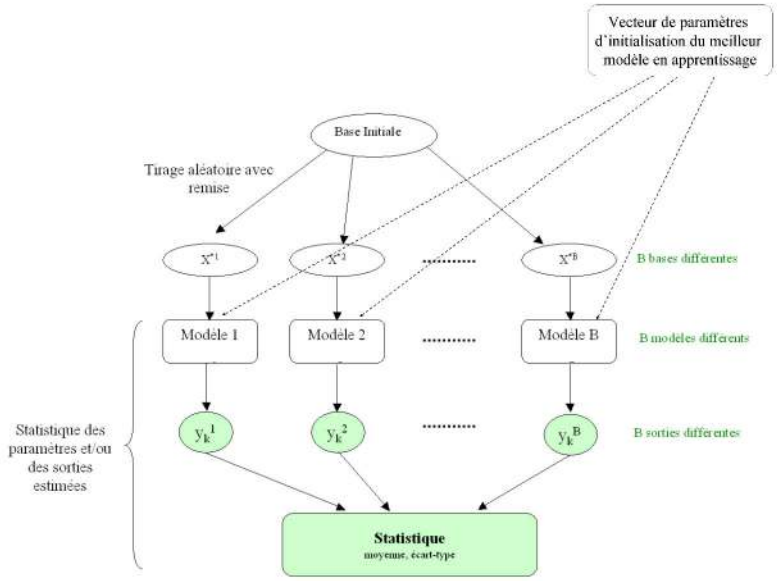



FIG. 3.13 – Utilisation du bootstrap pour déterminer la statistique des sorties estimées par le modèle. Les modèles obtenus à partir des répliques bootstrap sont initialisés avec le même vecteur de paramètres qui correspond au vecteur de paramètres initial du modèle, créé à partir de la base complète, qui a obtenu l'erreur d'apprentissage la plus faible.

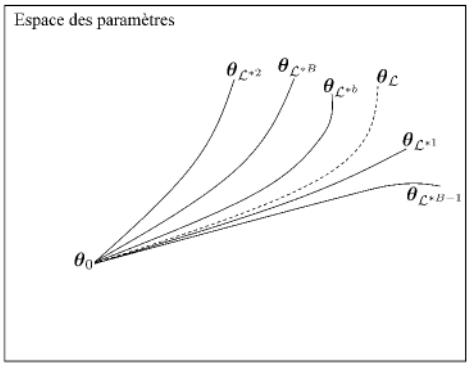



FIG. 3.14 – Le modèle correspondant aux pointillés est le meilleur en apprentissage parmi un grand nombre de modèles entrainés avec différentes initialisations. Son jeu de paramètres initial est utilisé pour initialiser tous les modèles dont la base d'apprentissage est une réplique bootstrap.

Si le modèle est non linéaire, en plus du choix de la complexité, nous devons déterminer le jeu de paramètres qui offre les meilleurs résultats en généralisation parmi d'autres modèles de même complexité. Etant donné que pour les benchmarks du dernier chapitre, nous serons dans le cadre de processus déterministes, on choisira le meilleur modèle sur ses capacités d'apprentissage.

Enfin, pour un modèle non linéaire, on peut utiliser des méthodes de régularisation pour éviter le surajustement, comme l'arrêt prématuré (" early stopping") que nous avons décrit au premier chapitre. Nous avons défini en début de chapitre comment estimer l'erreur de généralisation par Bootstrap. Afin de déterminer le nombre de cycles optimum, nous allons estimer l'erreur de généralisation à chaque cycle. Nous utilisons alors le résidu à l'iteration $i$ de l'exemple $k$ sur la réplique bootstrap $b$, noté $r^b_{i,k}$ [Martinez 04] :

$$r^b_{i,k} = y_k - f(\mathbf{x}^{*b}_k, \boldsymbol{\theta}^i_{\mathcal{L}^{*b}})$$

où $\boldsymbol{\theta}^i_{\mathcal{L}^{*b}}$, est le jeu de paramètres du modèle à l'iteration $i$, obtenu à l'aide des données de la replique $b$. Les erreurs de validation et d'apprentissage à l'itération $i$ sont définies de la manière suivante :

$$\begin{aligned} E^{*b}_a(i) &= \sqrt{\frac{1}{N}\sum_{k=1}^{N}(y^{*b}_k - f(\mathbf{x}^{*b}_k, \boldsymbol{\theta}^i_{\mathcal{L}^{*b}}))^2} \\ E^{*b}_v(i) &= \sqrt{\frac{1}{N}\sum_{k=1}^{N}(y_k - f(\mathbf{x}_k, \boldsymbol{\theta}^i_{\mathcal{L}^{*b}}))^2} \end{aligned}$$

Nous pouvons alors suivre l'estimation de l'erreur de généralisation estimée par bootstrap ($E^{*b}_v(i)$) en fonction du nombre de cycles d'apprentissage et déterminer le nombre de cycles $i$ pour lequel $E^{*b}_v(i)$ est minimum.

## 3.6 Conclusion

Dans ce chapitre, nous avons décrit quelques méthodes d'estimation de l'erreur de généralisation. Une première méthode permettant d'estimer l'espérance du risque fonctionnel, pour des modèles linéaires en leurs paramètres, nous a permis d'établir un lien entre l'apprentissage statistique et les plans d'expériences : une base d'apprentissage spécifiée par la A-optimalité revient à minimiser, dans ce cas, l'espérance du risque fonctionnel. Nous avons ensuite défini d'autres méthodes d'estimation de l'erreur de généralisation fondées sur des techniques de ré-échantillonnages comme le leave-one-out, la validation croisée et le Bootstrap. Ces méthodes d'estimation peuvent s'appliquer à des modèles non linéaires par rapport aux paramètres. Nous avons utilisé, à travers des fonctions tests, ces méthodes d'estimation de l'erreur de généralisation à la problèmatique de choix de complexité, pour des modèles linéaires par rapport aux paramètres. Enfin, nous avons traité la problématique du choix de modèle pour une architecture donnée dans le cadre des modèles non linéaires par rapport aux paramètres. L'heuristique que

nous utiliserons, dans la suite du mémoire, pour choisir le vecteur de paramètres le plus satisfaisant, est fondée sur l'observation des corrélations fortes existant entre l'erreur d'apprentissage et l'erreur de généralisation dans le cadre des données déterministes. Le jeu de paramètres initiaux du modèle, créé à partir de la base d'apprentissage complète, obtenant l'erreur d'apprentissage la plus faible, sera utilisé pour initialiser l'ensemble des modèles construits à partir de répliques bootstrap.

Le chapitre 5 décrit la méthode d'apprentissage actif que nous proposons et la compare à la méthode des plans optimaux. Le chapitre suivant décrit donc la méthodologie des plans d'expériences optimaux afin de mieux aborder cette comparaison.

# Chapitre 4

# Méthode des plans d'expériences

**Sommaire**


**4.1 Introduction . . . . . 115**
**4.2 Estimation des paramètres et prédiction . . . . . 116**
4.2.1 Modélisation . . . . . 116
4.2.2 Matrice de variance-covariance de $\boldsymbol{\theta}_{\mathcal{L}}$ . . . . . 117
4.2.3 La prédiction . . . . . 118
4.2.4 La fonction de variance de prédiction . . . . . 119
**4.3 Les Plans Optimaux pour modèle linéaire par rapport aux paramètres 120**
4.3.1 Conception . . . . . 120
4.3.2 Le critère de E-optimalité . . . . . 121
4.3.3 Le critère de A-optimalité . . . . . 121
4.3.4 Le critère de D-optimalité . . . . . 122
4.3.5 Le critère de G-optimalité . . . . . 123
4.3.6 Le critère de J-optimalité . . . . . 123
**4.4 Construction des plans optimaux . . . . . 125**
4.4.1 L'algorithme d'échange double de Fedorov . . . . . 125
4.4.2 Exemple d'utilisation d'un plan D-optimal . . . . . 126
4.4.3 Les leviers et la D-optimalité . . . . . 130
**4.5 Les plans optimaux pour modèles non linéaire par rapport aux paramètres . . . . . 130**
4.5.1 La D-optimalité locale . . . . . 130
4.5.2 La X-optimalité . . . . . 131
**4.6 Conclusion . . . . . 132**

## 4.1 Introduction

La mise en oeuvre des plans d'expériences peut avoir plusieurs objectifs :

- trouver, par un ensemble d'expériences, les facteurs qui ont une influence sur la grandeur à modéliser ("screening") ; à cette fin, on utilise des *plans de criblage* [Goupy 99], [Droesbeke *et al.* 97], [Box & Draper 87] comme les plans factoriels [Kobilinsky 97] ou les plans de Plackett-Burman [Droesbeke *et al.* 97]
- constituer un ensemble de données expérimentales en vue d'estimer les paramètres d'un modèle, linéaire ou non linéaire.

À titre d'exemple, supposons que l'on cherche à constituer un modèle prédictif du rendement d'une réaction chimique faisant intervenir deux facteurs continus $(x_1, x_2)$. Ne disposant pas de modèle théorique pour cette expérience, on postule un modèle polynomial de degré deux.

$$f(\mathbf{x}, \boldsymbol{\theta}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2 + \epsilon$$

où les $x_i$ sont les facteurs et où les paramètres $\theta_i$ traduisent les effets des ces facteurs. Pour ce modèle postulé et pour le domaine expérimental $[-1; 1]^2$, un plan d'expériences possible, orthogonal[12] serait constitué de 9 points disposés comme indiqué sur la figure 4.1.

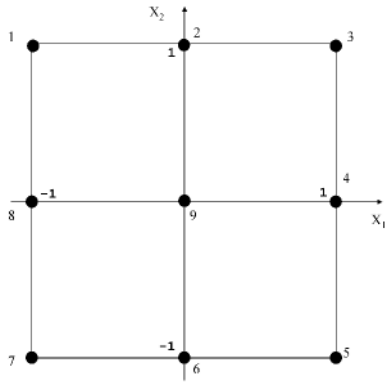



Fig. 4.1 – Domaine expérimental sans contraintes

Les contraintes expérimentales peuvent être de plusieurs types. Par exemple, certaines réactions peuvent ne pas être réalisables ; dans ce cas, le domaine expérimental est un domaine expérimental sous contraintes comme le montre la figure 4.2.

Certaines expériences n'étant pas réalisables, le plan ne contient que six expériences. La qualité du plan d'expériences est dégradée. Il y a six expériences pour estimer six paramètres (aucun degré de liberté ne subsiste). Il faut donc trouver d'autres expériences dans ce domaine sous contrainte pour améliorer la qualité du plan d'expériences en un sens que nous préciserons.

[12] A condition que l'on transforme les termes quadratiques par codage avec les polynômes orthogonaux.

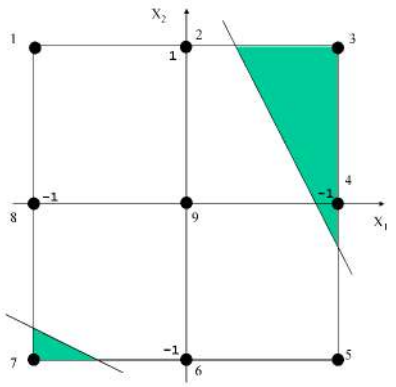



FIG. 4.2 – Domaine expérimental sous contraintes

Le modèle postulé n'est pas forcément empirique, comme nous l'avons fait dans l'exemple précédent : il se peut que l'on dispose d'un modèle fondé sur les connaissances concernant le processus à modéliser ; ce modèle peut faire intervenir des paramètres physiques ou physico-chimiques dont on cherche à estimer les valeurs numériques à partir d'expériences.

Pour ces deux familles de problème, on cherche à réaliser un plan d'expériences "optimal" au sens d'un critère de qualité approprié ; nous décrirons dans ce chapitre plusieurs critères d'optimalité pour les plans d'expériences.

Comme nous l'avons souligné dans l'introduction générale, nous voulons approcher des codes de calcul par des surfaces de réponse neuronales. Leur construction nécessite également des plans d'expériences afin d'obtenir ces méta-modèles par régression non linéaire. Nous proposons au dernier chapitre, la méthode d'apprentissage actif LDR. Nous voulons, dans le cadre de cette thèse, comparer cette méthode à des méthodes déjà existantes comme les plans d'expériences optimaux. Nous décrirons tout d'abord les principaux plans d'expériences optimaux pour modèles linéaires par rapport aux paramètres, puis nous aborderons la construction de plans d'expériences pour modèles non linéaires par rapport aux paramètres.

## 4.2 Estimation des paramètres et prédiction

### 4.2.1 Modélisation

Rappelons que les $N$ expériences d'un plan d'expériences sont contenues dans la matrice $\mathbf{L}_N$

$$\mathbf{L}_N = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1h} \\ x_{21} & x_{22} & \dots & x_{2h} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{Nh} \end{pmatrix}$$

qui contient autant de lignes que d'exemples et autant de colonnes que de facteurs. L'élément $x_{ij}$ est la valeur du facteur $j$ de l'exemple $i$. En passant aux variables secondaires à l'aide des fonctions $\phi_i(\mathbf{x})$ on définit la matrice $\mathbf{X}$ de dimension ($N$,$p$) appelée matrice du modèle ou matrice des effets.

Elle contient $N$ lignes correspondant au nombre d'exemples du plan d'expériences et le nombre $p$ de colonnes est égal au nombre de paramètres du modèle. L'élément $ij$ de cette matrice du modèle est la valeur du régresseur $j$ pour l'exemple $i$ ($\phi_j(\mathbf{x}_i)$).

$$\mathbf{X} = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \dots & \phi_p(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \dots & \phi_p(\mathbf{x}_2) \\ \vdots & \vdots & \phi_j(\mathbf{x}_i) & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \dots & \phi_p(\mathbf{x}_N) \end{pmatrix}$$

Le modèle linéaire s'écrit :

$$f(\mathbf{x}, \boldsymbol{\theta}) = \boldsymbol{\phi}(\mathbf{x})^{'}\boldsymbol{\theta}$$

On minimise la fonction de coût des moindres carrés :

$$\begin{aligned} J(\boldsymbol{\theta}) &= \frac{1}{N}\sum_{k=1}^{N}(y_k - f(\mathbf{x}_k, \boldsymbol{\theta}))^2 \\ &= \frac{1}{N}||\mathbf{X}\boldsymbol{\theta} - \mathbf{y}||^2 \end{aligned}$$

En supposant la matrice d'information ($\mathbf{X'X}$) inversible, l'estimation $\boldsymbol{\theta}_{\mathcal{L}}$ des paramètres $\boldsymbol{\theta}$ par moindres carrés est donnée par :

$$\boldsymbol{\theta}_{\mathcal{L}} = (\mathbf{X'X})^{-1}\mathbf{X'y}$$

Le vecteur $\mathbf{y}$ étant modélisé comme une réalisation d'une variable aléatoire, l'estimation $\boldsymbol{\theta}_{\mathcal{L}}$ l'est également. Nous allons, à présent, définir la matrice de variance-covariance de $\boldsymbol{\theta}_{\mathcal{L}}$.

### 4.2.2 Matrice de variance-covariance de $\boldsymbol{\theta}_{\mathcal{L}}$

Sous les hypothèses que les facteurs sont des variables certaines et que l'erreur expérimentale est centrée, non corrélée de variance $\sigma^2$, la matrice de variance-covariance des paramètres s'exprime comme :

$$s^2(\boldsymbol{\theta}_{\mathcal{L}}) = s^2((\mathbf{X'X})^{-1}\mathbf{X'y}) = (\mathbf{X'X})^{-1}\mathbf{X'}s^2(\mathbf{y})\mathbf{X}(\mathbf{X'X})^{-1} = \sigma^2(\mathbf{X'X})^{-1} \tag{4.1}$$

La variance de la composante $j$ du vecteur $\boldsymbol{\theta}_{\mathcal{L}}$, notée $\boldsymbol{\theta}_{\mathcal{L},j}$ est donc le produit de la variance de l'erreur expérimentale $\sigma^2$ par le terme diagonal de la matrice de dispersion $(\mathbf{X}'\mathbf{X})^{-1}$, inverse de la matrice d'information de Fisher.

$$s^2([\theta_{\mathcal{L},j}]) = c_{jj}\sigma^2$$

De même, la covariance $(\theta_{\mathcal{L},i}, \theta_{\mathcal{L},j})$ est le produit de la variance de l'erreur expérimentale $\sigma^2$ par le terme correspondant de la matrice de dispersion.

$$Covar[\theta_{\mathcal{L},i}, \theta_{\mathcal{L},j}] = c_{ij}\sigma^2$$

La variance des paramètres du modèle ne dépend donc que :

- de la variance de l'erreur expérimentale $\sigma^2$
- des éléments de la matrice de dispersion $(\mathbf{X}'\mathbf{X})^{-1}$, donc des positions des points expérimentaux dans le domaine d'étude et du choix des fonction $\phi_i$.

Les plans d'expériences, c'est-à-dire les stratégies de choix des points expérimentaux, peuvent donc avoir pour objectif la conception de modèles pour lesquels les estimateurs des paramètres ont une variance aussi faible que possible.

### 4.2.3 La prédiction

Après avoir estimé les paramètres, on peut prédire la réponse en un point $I$ quelconque du domaine, par :

$$f(\mathbf{x}_i, \boldsymbol{\theta}_{\mathcal{L}}) = \phi(\mathbf{x}_i)' \boldsymbol{\theta}_{\mathcal{L}}$$

La variance de l'estimation peut également constituer un critère de qualité du modèle. Elle a pour expression :

$$s^2(f(\mathbf{x}_i, \boldsymbol{\theta}_{\mathcal{L}})) = s^2(\phi(\mathbf{x}_i)' \boldsymbol{\theta}_{\mathcal{L}}) = \phi(\mathbf{x}_i)' s^2(\boldsymbol{\theta}_{\mathcal{L}}) \phi(\mathbf{x}_i) = \phi(\mathbf{x}_i)' (\mathbf{X}'\mathbf{X})^{-1} \phi(\mathbf{x}_i) \sigma^2 = h_{ii}\sigma^2 \qquad (4.2)$$

La quantité $h_{ii}$ est le levier de l'exemple $i$ ; dans le formalisme de la théorie des plans d'expériences, il est également appelé fonction de variance de prédiction au point $\mathbf{x}_i$. La variance de la réponse du modèle en un point est donc proportionnelle au levier de ce point. On verra par la suite que la G-optimalité vise à construire des plans d'expériences de façon à réduire le maximum des variances de prédiction.

### 4.2.4 La fonction de variance de prédiction

Comme nous l'avons indiqué dans l'introduction générale, nous utiliserons, dans la méthode LDR, la variance de prédiction, afin de planifier des expériences dans le cadre de modèles non linéaires par rapport aux paramètres. Nous décrirons cette méthode dans le dernier chapitre. Il est intéressant d'estimer la variance des réponses prédites en un point du domaine expérimental. Par exemple, on peut utiliser cette fonction pour vérifier le bon ajustement du modèle postulé. En effet, la fonction de variance de prédiction s'exprime de la manière suivante :

$$d^2(f(\mathbf{x}_p, \boldsymbol{\theta}_{\mathcal{L}})) = \phi(\mathbf{x}_p)^{'}(\mathbf{X'X})^{-1}\phi(\mathbf{x}_p) \tag{4.3}$$

En général, on préfère utiliser la racine carrée de la fonction de variance de prédiction (ou fonction d'erreur de prédiction), car elle s'exprime dans la même unité que la réponse.

$$d(f(\mathbf{x}_p, \boldsymbol{\theta}_{\mathcal{L}})) = \sqrt{\phi(\mathbf{x}_p)'(\mathbf{X'X})^{-1}\phi(\mathbf{x}_p)}$$

Pour des modèles postulés linéaires par rapport aux paramètres, la fonction d'erreur de prédiction ne dépend donc que de l'emplacement des points expérimentaux et du modèle postulé ; elle est indépendante du résultat des expériences. On peut donc savoir, avant de commencer les expériences, dans quelle mesure le modèle et l'emplacement des points expérimentaux affecteront la précision des réponses prédites. La variance de la réponse prédite peut donc s'écrire [Goupy 99] :

$$s^2(f(\mathbf{x}_p, \boldsymbol{\theta}_{\mathcal{L}})) = \sigma^2\phi(\mathbf{x}_p)^{'}(\mathbf{X'X})^{-1}\phi(\mathbf{x}_p) = \sigma^2 d^2(f(\mathbf{x}_p, \boldsymbol{\theta}_{\mathcal{L}}))$$

d'où

$$d^2(f(\mathbf{x}_p, \boldsymbol{\theta}_{\mathcal{L}})) = \frac{s^2(f(\mathbf{x}_p, \boldsymbol{\theta}_{\mathcal{L}}))}{\sigma^2}$$

soit encore :

$$d(f(\mathbf{x}_p, \boldsymbol{\theta}_{\mathcal{L}})) = \frac{s(f(\mathbf{x}_p, \boldsymbol{\theta}_{\mathcal{L}}))}{\sigma}$$

La fonction d'erreur de prédiction est égale au rapport de l'écart-type de la réponse prédite à l'écart-type du bruit expérimental. Si nous considérons que le modèle postulé est bien ajusté lorsque l'écart-type de prédiction se confond avec le bruit expérimental, alors le modèle postulé sera considéré bien ajusté si la fonction d'erreur de prédiction est inférieure à l'unité.

## 4.3 Les Plans Optimaux pour modèle linéaire par rapport aux paramètres

Nous allons décrire, dans ce paragraphe, les plans d'expériences "optimaux", pour lesquels les expériences sont choisies afin de satisfaire un critère d'optimalité. Nous allons tout d'abord rappeler l'origine des plans d'expériences optimaux avant de décrire les principaux critères d'optimalité utilisés, ainsi que la manière de construire ces plans. Ce paragraphe est largement inspiré de [Gauchi 97a].

### 4.3.1 Conception

Un exposé complet exigerait la présentation des mesures de probabilité continues sur l'espace expérimental, nous renvoyons le lecteur à [Gauchi 97a] pour une introduction sur cet aspect. Nous nous limiterons ici à exposer les éléments relatifs aux plans optimaux discrets (donc physiquement ou numériquement réalisables). Comme nous l'avons vu dans l'équation (2.23) du paragraphe 4.2.2, la variance des paramètres s'exprime de la manière suivante :

$$s^2(\boldsymbol{\theta}_{\mathcal{L}}) = \sigma^2(\mathbf{X}'\mathbf{X})^{-1}$$

La variance des paramètres estimés dépend donc des propriétés spectrales de la matrice de dispersion $(\mathbf{X}'\mathbf{X})^{-1}$. La valeur de $\sigma^2$ étant imposée par le problème, il est possible de chercher la distribution de points expérimentaux qui permet de minimiser la variance des paramètres estimés.

Sous l'hypothèse d'un bruit gaussien et d'un modèle linéaire de matrice d'information de Fisher $\mathbf{X}'\mathbf{X}$, la région de confiance de niveau $1-\alpha$ de l'estimation $\boldsymbol{\theta}_{\mathcal{L}}$ des $p$ paramètres du modèle est un hyperellipsoïde défini dans [Antoniadis *et al.* 92], [Rao & Toutenburg 95], centré sur le vecteur $\boldsymbol{\theta}_{\mathcal{L}}$ et dont la frontière est définie par l'inégalité suivante (si $\sigma^2$ est connue :

$$(\boldsymbol{\theta} - \boldsymbol{\theta}_{\mathcal{L}})'\mathbf{X}'\mathbf{X}(\boldsymbol{\theta} - \boldsymbol{\theta}_{\mathcal{L}}) \leq \sigma^2\chi^2_{\alpha}(p) \tag{4.4}$$

où $\chi^2_{\alpha}(p)$ est le fractile $\alpha$ de la loi du Chi-2 à p degrés de liberté.

Pour réduire la variance des paramètres estimés, il faut donc agir sur les propriétés spectrales de la matrice d'information de Fisher afin de modifier le volume, la forme et les directions des

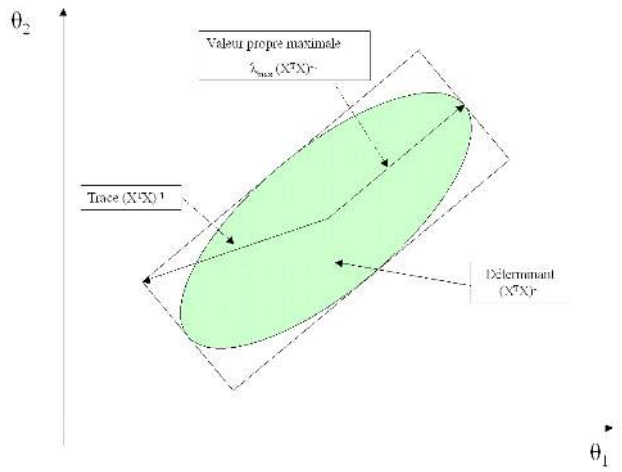



FIG. 4.3 – Zone de confiance des paramètres estimés

axes de cet ellipsoïde. Il existe plusieurs façons d'agir sur les propriétés spectrales de la matrice de Fisher, certaines d'entre elles correspondent aux critères d'optimalité que nous allons décrire.

### 4.3.2 Le critère de E-optimalité

Le critère de E-optimalité (E comme "eigen value") porte sur la longueur du plus grand axe de l'ellipse. Pour minimiser celle-ci, on minimise la plus grande valeur propre de $(\mathbf{X'X})^{-1}$, ou, de manière équivalente, on maximise la plus petite valeur propre de $\mathbf{X'X}$. Désignons par $L_N^E$ un plan E-optimal comprenant $N$ expériences :

$$L_N^E = Arg\{min_{L_N \in \Pi}[max_i(\frac{1}{\gamma_i})]\} \tag{4.5}$$

où $L_N$ est un plan de $N$ expériences, $\Pi$ est l'ensemble des plans d'expériences possibles, et les $\gamma_i$ sont les valeurs propres de la matrice de Fisher $\mathbf{X'X}$. La détermination d'un plan E-optimal est une tâche numériquement lourde car elle nécessite le calcul répété des valeurs propres de $\mathbf{X'X}$.

### 4.3.3 Le critère de A-optimalité

Le critère de A-optimalité (A pour "Average optimal") porte sur la valeur de la trace de $(\mathbf{X'X})^{-1}$. En souhaite réduire la longueur de la diagonale du pavé exinscrit à l'hyperellipsoïde en tenant compte de l'aplatissement de l'ellipsoïde. On minimise donc la somme des variances des paramètres. Désignons par $L_N^A$ un plan A-optimal comprenant $N$ expériences :

$$L_N^A = Arg\{min_{P_N \in \Pi}[Tr((\mathbf{X'X})^{-1})]\} \tag{4.6}$$

Le calcul d'un plan A-optimal est moins coûteux en temps de calcul que celui d'un plan E-optimal, mais il est plus lourd que le calcul d'un plan D-optimal, que nous verrons au paragraphe suivant, car il faut inverser $(\mathbf{X}'\mathbf{X})$. Remarquons également que le critère de A-optimalité est invariant pour toute transformation orthogonale $\Gamma$ sur les paramètres :

$$Tr(\mathbf{\Gamma}^{'}\mathbf{Q}\mathbf{\Gamma}) = Tr(\mathbf{Q}\mathbf{\Gamma}\mathbf{\Gamma}^{'}) = Tr(\mathbf{Q})$$

C'est ce critère qui permet de réduire le rapport $T(d, \mathcal{L}_N)$ de l'espérance du risque fonctionnel à l'espérance du risque empirique, que nous avons décrit au paragraphe 3.2 :

$$T(d, \mathcal{L}_N) = (1 - \frac{d}{N})^{-1}(1 + \frac{\sum_{i=1}^{d} 1/\gamma_i}{N})$$

où $d$ est le nombre de régresseurs, $\mathcal{L}_N$ est le plan de $N$ expériences, et les $\gamma_i$ sont les valeurs propres de la matrice d'information de Fisher.

### 4.3.4 Le critère de D-optimalité

Enfin, on peut choisir un critère qui revient à minimiser le volume de l'ellipsoïde. En minimisant ce volume, on optimise le produit des variances des paramètres. On minimise donc le déterminant de la matrice de dispersion (produit des valeurs propres) ou, de manière équivalente, on maximise le déterminant de la matrice de Fisher. Désignons par $L_N^D$ un plan D-optimal comprenant $N$ expériences :

$$L_N^D = Arg\{max_{P_N \in \Pi}[det(\mathbf{X}'\mathbf{X})]\} \tag{4.7}$$

Notons l'invariance de ce critère pour toute transformation de matrice $\mathbf{T}$ des fonctions de régression ($\psi(\mathbf{x}) = \mathbf{T}\phi(\mathbf{x})$) à condition que $det(\mathbf{L})$ soit non nul, ou pour une reparamétrisation. Dans le cas, par exemple, d'une reparamétrisation, $f(\mathbf{X}) = \mathbf{X}\boldsymbol{\theta} = \mathbf{Z}\boldsymbol{\beta}$ avec $\mathbf{Z} = \mathbf{X}\mathbf{U}$, et $\mathbf{U}$ ne dépendant pas du plan, alors :

$$det(\mathbf{Z}'\mathbf{Z}) = det(\mathbf{U}'\mathbf{X}'\mathbf{X}\mathbf{U}) = det(\mathbf{X}'\mathbf{X})det(\mathbf{U})^2 \tag{4.8}$$

Donc si $\mathbf{X}_1$ est meilleur que $\mathbf{X}_2$ au sens de la D-optimalité, alors $\mathbf{Z}_1 = \mathbf{X}_1\mathbf{U}$ est meilleur que $\mathbf{Z}_2 = \mathbf{X}_2\mathbf{U}$. Autrement dit, un plan D-optimal est invariant pour toute transformation linéaire des régresseurs et des paramètres comme lors d'un changement d'échelle.

Enfin, il est préférable d'utiliser le déterminant normé pour comparer deux plans d'expériences qui n'auraient pas le même nombre d'exemples ; la valeur du déterminant dépend du

nombre d'exemples que contient le plan d'expériences. Le déterminant normé permet de s'affranchir de l'influence du nombre d'exemples et permet donc d'estimer véritablement la qualité du plan au sens du critère de la D-optimalité. Notons par $det_{\mathcal{N}}(\mathbf{X}'\mathbf{X})$ ce déterminant normé :

$$det_{\mathcal{N}}(\mathbf{X}'\mathbf{X}) = det(\frac{1}{N}\mathbf{X}'\mathbf{X}) = \frac{1}{N^p}det(\mathbf{X}'\mathbf{X})$$

où $p$ est le nombre de paramètres du modèle.

### 4.3.5 Le critère de G-optimalité

Pour les trois critères que nous venons de voir, on cherche un plan qui permet d'estimer $\hat{\boldsymbol{\theta}}$ avec la plus grande confiance. On peut envisager d'utiliser également un critère prennant en considération, non pas la confiance dans l'estimation des paramètres, mais la variance des prédictions du modèle. C'est le critère de G-optimalité. On utilise alors la fonction de variance de prédiction (4.3) vue au paragraphe 4.2.4 :

$$d^2(f(\mathbf{x}_p, \boldsymbol{\theta}_{\mathcal{L}})) = \phi(\mathbf{x}_p)'(\mathbf{X}'\mathbf{X})^{-1}\phi(\mathbf{x}_p)$$

Le plan G-optimal vise alors à minimiser le maximum de la fonction de variance de prédiction :

$$L_N^G = Arg\{min[max_{x_i \in U} d^2(f(\mathbf{x}_i, \boldsymbol{\theta}_{\mathcal{L}}))]\} \tag{4.9}$$

où $U$ est le domaine d'étude.

Rappelons que la fonction de prédiction en $\mathbf{x}$ est le levier de l'exemple $\mathbf{x}$ dans le monde de l'apprentissage statistique, comme nous l'avons vu aux chapitres 2 et 3. Nous verrons au dernier chapitre, que la méthode d'apprentissage actif que nous proposons est proche du critère de G-optimalité. La variance de prédiction est estimée par la méthode de ré-échantillonnage du Bootstrap dans la méthode LDR.

### 4.3.6 Le critère de J-optimalité

Le critère de J-optimalité, appelé aussi IMSE en anglais (Integrated Mean Square Error), a été proposé par [Box & Draper 59]. L'originalité et l'intérêt de ce critère est qu'il prend en considération à la fois la variance de prédiction de la réponse et le biais du modèle postulé lorsque celui-ci n'est pas le "bon" modèle. Ce critère consiste à minimiser par rapport à la mesure

$\pi$ d'un plan à mesure continue[13] la quantité J :

$$J = \frac{ND}{\sigma^2} \int_U E[f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}}) - y(\mathbf{x})]^2 d\mathbf{x} \tag{4.10}$$

où :
- N est le nombre total d'expériences du plan,
- $D^{-1} = \int_U d\mathbf{x}$,
- $f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}})$ la prédiction du modèle postulé,
- $y(\mathbf{x})$ la réponse du modèle de régression.

Sans entrer dans le détail de ce critère (car il prend en considération les mesures de plans continus qui ne rentrent pas dans le cadre de notre étude, pour plus d'information voir [Gauchi 05]), il peut se décomposer en "variance+biais" $J = V + B$ où :

$$V = \frac{ND}{\sigma^2} \int_U s^2(f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}}))d\mathbf{x} = \frac{ND}{\sigma^2} E(\sigma^2(f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}})))$$

$s^2(f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}})$ peut être obtenue par la fonction de variance de prédiction (4.3).

$$B = \frac{ND}{\sigma^2} \int_U \left[E[f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}})] - y(\mathbf{x})\right]^2 d\mathbf{x}$$

La démarche que nous utiliserons pour planifier des points par apprentissage actif se rapproche également du problème de minimisation du biais entre le modèle *a priori* et le vrai modèle (par les méthodes d'apprentissage décrites au chapitre 2) et la planification de nouvelles expériences afin de réduire la variance des prédictions (estimée par bootstrap). Nous analyserons cela au chapitre suivant, qui traite de la méthode LDR que nous proposons.

Nous pouvons remarquer que la théorie des plans d'expériences est fondée sur l'existence d'un bruit expérimental, comme nous l'avons décrit au paragraphe 4.3.1. C'est l'existence de ce bruit expérimental qui justifie d'agir sur les propriétés spectrales de la matrice de Fisher pour changer le volume, la forme et la direction des axes de l'ellipsoïde de confiance. Cependant, nous pouvons remarquer que le bruit de mesure n'intervient pas dans les différents critères d'optimalité que nous avons décrit. Il est alors tout à fait possible de construire des plans optimaux en considérant la variance du bruit expérimental comme nulle. C'est en prenant en considération cette remarque que nous pourrons comparer la méthode que nous proposons au dernier chapitre avec les plans d'expériences numériques optimaux pour modèles non linéaires, comme, par exemple, la D-optimalité locale que nous verrons à la fin de ce chapitre.

[13] Un plan à mesure continue a été introduit par Kiefer. On considère le plan d'expériences avec une mesure uniforme de probabilité des points $\mathbf{x}$ (design mesure [Gauchi 05]).

## 4.4 Construction des plans optimaux

Il existe beaucoup d'algorithmes pour la construction de plans d'expériences optimaux [Wynn 70], [Mitchell 74] ou [Atkinson & Donev 89]. Nous allons décrire l'algorithme d'échange double de Fedorov [Fedorov 72] qui est probablement le plus utilisé et le plus facile à implémenter. Nous présentons cet algorithme dans le cas de la construction d'un plan D-optimal, mais on peut l'utiliser pour satisfaire d'autres critères que celui de la D-optimalité comme, par exemple, ceux de la A- ou de la E-optimalité.

### 4.4.1 L'algorithme d'échange double de Fedorov

L'objectif de l'algorithme est de sélectionner $N$ expériences parmi les $N_c$ expériences candidates qui sont les noeuds d'une grille résultat de la discrétisation du domaine d'étude. Il faut trouver, parmi tous les plans $L_N$ possibles, le plan d'expériences $L_N^D$ de $N$ expériences pour lequel le déterminant de la matrice d'information est le plus élevé. Pour obtenir ce plan[14], on procède en trois étapes.

- Etape 1 : pour initialiser l'algorithme, on choisit de manière aléatoire un jeu de $N$ expériences parmi les $N_c$ expériences candidates.

- Etape 2 : on remplace l'expérience $i$ du plan de $N$ expériences par l'expérience $j$ de l'ensemble des expériences candidates. On veut trouver le couple $(i_{max}, j_{max})$ qui permet d'obtenir le plus grand accroissement du déterminant de la matrice de Fisher ; pour cela, on calcule les $N \times (N_c - 1)$ déterminants possibles si l'on autorise les répétitions d'expériences, ou les $N \times (N_c - N)$ déterminants possibles si l'on n'autorise pas les répétitions ; cette dernière situation se présente dans le cas des expériences numériques, où la répétition ne donne aucune information supplémentaire.

- Etape 3 : on introduit l'expérience $j_{max}$ dans le plan d'expériences et l'on en retire l'expérience $i_{max}$. On retourne ensuite à l'étape 2 jusqu'à ce que le critère d'arrêt soit vérifié.

On peut arrêter l'algorithme, par exemple, lorsque l'accroissement du déterminant est plus petit qu'un $\epsilon$ fixé *a priori*.

Pour calculer le déterminant à l'itération courante, on peut utiliser le théorème que l'on peut trouver dans [Rao & Toutenburg 95] et qui est repris dans [Gauchi 05].

L'échange d'une expérience $i$ et d'une expérience $j$ se traduit, dans la matrice d'information, par l'introduction d'un vecteur colonne $\mathbf{x}_j$ à la place du vecteur colonne $\mathbf{x}_i$. La nouvelle matrice d'information est :

$$(\mathbf{X'X})_{[t+1]} = (\mathbf{X'X})_{[t]} - \phi(\mathbf{x}_i)\phi(\mathbf{x}_i)' + \phi(\mathbf{x}_j)\phi(\mathbf{x}_j)' \tag{4.11}$$

[14] Il n'y a pas de preuve formelle de convergence, mais l'algorithme converge très souvent vers le maximum global [Gauchi 97b].

Par conséquent, le déterminant est :

$$det((\mathbf{X}'\mathbf{X})_{[t+1]}) = det((\mathbf{X}'\mathbf{X})_{[t]}) \times [1 + \Delta(i,j)] \tag{4.12}$$

avec

$$\Delta(i,j) = h_{jj} - [h_{ii}h_{jj} - h_{ij}^2] - h_{ii}$$

où $h_{ij}$ est l'élément $ij$ de la matrice chapeau $H_t = [\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}']_t$.

$$h_{ij} = \phi(\mathbf{x}_i)'(\mathbf{X}'\mathbf{X})_t^{-1}\phi(\mathbf{x}_j) \tag{4.13}$$

Dans le cadre de ce mémoire, nous utilisons des plans optimaux discrets[15], pour lesquels il n'y a pas unicité du résultat. L'optimalité du plan dépend notamment de la grille de points candidats utilisée et du tirage aléatoire du premier plan de l'étape 1. Étant donné qu'il n'y a pas unicité du résultat, il est préferable de relancer $k$ fois l'algorithme en partant de $k$ tirages aléatoires initiaux différents. En pratique, $k$ est compris entre 5 et 10 et l'on choisit le meilleur plan parmi les $k$ plans créés.

### 4.4.2 Exemple d'utilisation d'un plan D-optimal

Á titre d'illustration, nous présentons ici un exemple très simple : le placement de $N$ points pour un modèle linéaire à un facteur.

$$y_i = \theta_0 + \theta_1 x_i + \epsilon_i \qquad x_i \in [-1;1] \qquad \forall i \in [1,N]$$

La matrice du modèle $\mathbf{X}$ est :

$$\mathbf{X} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & \vdots \\ 1 & x_N \end{pmatrix}$$

D'où la matrice d'information

$$\mathbf{X}'\mathbf{X} = \begin{pmatrix} N & \sum_{i=1}^N x_i \\ \sum_{i=1}^N x_i & \sum_{i=1}^N x_i^2 \end{pmatrix}$$

[15]Comme nous l'avons décrit en début de paragraphe, les points candidats sont les noeuds d'un maillage correspondant à la discrétisation du domaine d'étude.

dont le déterminant est égal à :

$$det(\mathbf{X}'\mathbf{X}) = N\sum_{i=1}^{N} x_i^2 - (\sum_{i=1}^{N} x_i)^2$$

Maximiser le déterminant de $(X'X)$ revient à minimiser la fonction :

$$J(x) = (\sum_{i=1}^{N} x_i)^2 - N\sum_{i=1}^{N} x_i^2$$

sous les contraintes :

$$|x_i| \leq 1$$

Tout d'abord remarquons que $J(x) \geq -N^2$. En prenant $x_i = (-1)^i$ on obtient :

$$J(x) = \left\{ \begin{array}{cccc} -N^2 & \texttt{si} & N & \texttt{pair} \\ 1-N^2 & \texttt{si} & N & \texttt{impair} \end{array} \right.$$

Le résultat est donc obtenu pour $N$ pair puisque la borne inférieure est atteinte. Les deux égalités doivent être vérifiées : $\sum_i x_i^2 = N \Rightarrow |x_i| = 1$ et $\sum_i x_i = 0$, la moitié des valeurs se situe en 1 et l'autre moitié en $-1$. Le plan d'expériences D-optimal d'un nombre pair d'expériences consiste donc à spécifier la moitié des points sur le bord $x = -1$, l'autre moitié sur le bord $x = +1$.

Pour le cas $N$ impair, utilisons les multiplicateurs de Lagrange. On range les contraintes en contraintes actives (pour lequelles $\lambda_i$ est non nul) et les contraintes inactives (pour lesquelles $\lambda_i$ est nul). On note $F_i(\mathbf{x})$ la contrainte $x_i^2 \leq 1$, $q$ les contraintes actives et $\nabla$ l'opérateur dérivée :

$$\nabla J(\mathbf{x}) + \sum_{i=1}^{q} \lambda_i \nabla F_i(\mathbf{x}) = 0$$

À partir de $F_i(\mathbf{x}) = (0, \ldots, 2x_i, 0, \ldots)$, et en posant $a = \sum_i x_i$, on obtient les $N$ égalités :

$$\begin{array}{lllllll} 2a - 2Nx_i & +2\lambda_i x_i & = & 0 & 1 \leq & i & \leq q \\ 2a - 2Nx_i & & = & & q < & i & \leq N \end{array}$$

On en déduit $(\lambda_i - N)^2 = a^2$, $\texttt{pour}\ 1 \leq i \leq q$ et $x_i = a/N$, $\texttt{pour}\ q < i \leq N$. L'expression de la valeur minimale de $J$ est

$$\begin{aligned} J(x^*) &= -N[\sum_{i=1}^{q} x_i^2 + \sum_{i=q+1}^{N} x_i^2] + a^2 \\ &= -N[q + (N-q)\frac{a^2}{N^2}] + a^2 \\ &= -Nq + \frac{q}{N}a^2 \end{aligned}$$

D'autre part :

$$\begin{aligned} a &= \sum_{i=1}^{q} x_i + \sum_{i=q+1}^{N} x_i \\ &= \sum_{i=1}^{q} x_i + (N-q)\frac{a}{N} \\ &= \sum_{i=1}^{q} x_i + a - \frac{q}{N}a \end{aligned}$$

D'où $aq/N = \sum_{i=1}^{q} x_i = k$, nombre entier correspondant à la somme de $q$ nombres $+1$ ou $-1$. On a alors $J$ en fonction du nombre de contraintes actives $q$ et de leur répartition $k$ :

$$J(q,k) = -Nq + \frac{N}{q}k^2$$

$$\begin{aligned} q \texttt{ pair} &\Rightarrow k \in [-q, -q+2, \ldots, -2, 0, 2, \ldots, q] \\ q \texttt{ impair} &\Rightarrow k \in [-q, -q+2, \ldots, -1, 1, \ldots, q] \end{aligned}$$

Pour un nombre $q$ donné, la valeur minimale de $J$ est $k = 0$ si $q$ est pair et $k^2 = 1$ si $q$ est impair.

$$\begin{aligned} q \texttt{ pair} &\Rightarrow J \geq -Nq \\ q \texttt{ impair} &\Rightarrow J \geq -Nq + \frac{N}{q} \end{aligned}$$

Considérons le cas $q$ pair :

$$\begin{array}{rcccccc} N \texttt{ pair} & \Rightarrow & q = N & \Rightarrow & J^* & = & -N^2 \\ N \texttt{ impair} & \Rightarrow & q = N-1 & \Rightarrow & J^* & = & -N^2 + N \end{array}$$

Puis le cas $q$ impair

$$\begin{array}{rcccccc} N \texttt{ pair} & \Rightarrow & q = N-1 & \Rightarrow & J^* & = & -N(N-1) + \frac{N}{N-1} \\ N \texttt{ impair} & \Rightarrow & q = N & \Rightarrow & J^* & = & -N^2 + 1 \end{array}$$

L'examen de ces 4 cas montre que la solution minimale est atteinte pour $q = N$ quelle que soit la parité de $N$. La valeur maximale du déterminant est donc $N^2$ si $N$ est pair et $N^2 - 1$ si $N$ est impair. La solution optimale correspond donc aux contraintes actives pour l'ensemble des variables $x_i$. Lorsque $q = N$ dans le cas $N$ pair, on a $k = 0 \Rightarrow a = 0$ et donc les expériences $x_i$ doivent être partagées sur les valeurs $+1$ et $-1$. Pour $N$ impair, on a $a = 1$ ou $a = -1$, $N - 1$ expériences $x_i$ se partagent sur les valeurs $+1$ et $-1$, et la restante sur $+1$ ou $-1$.

Nous avions souligné qu'il fallait utiliser le déterminant normé pour comparer deux plans optimaux contenant un nombre différent d'expériences :

$$det_{\mathcal{N}}(\mathbf{X}'\mathbf{X}) = \frac{1}{N^p} det(\mathbf{X}'\mathbf{X})$$

où $p$ désigne le nombre de paramètres du modèle.

Sur cet exemple à deux paramètres, si $N$ est pair, le déterminant vaut $N^2$, par conséquent, le déterminant normé vaut 1. Si $N$ est impair, le déterminant vaut $N^2-1$, le déterminant normé vaut alors $1-\frac{1}{N^2}$.

La figure 4.4 représente la valeur du déterminant normé en fonction du nombre $N$ d'expériences du plan.

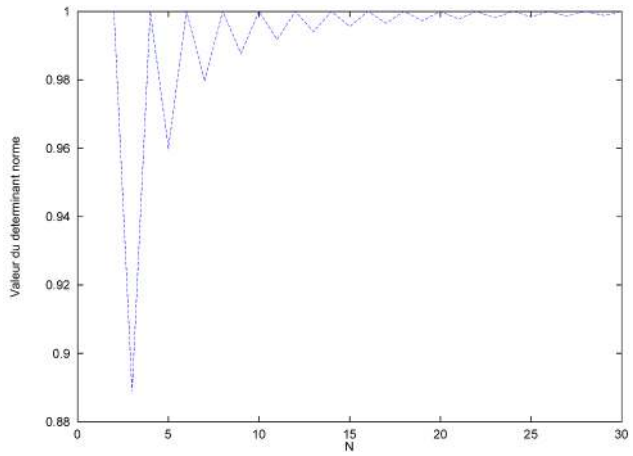



FIG. 4.4 – Valeur du déterminant normé en fonction du nombre $N$ d'expériences du plan.

On pourrait penser que plus il y a d'expériences dans le plan, plus l'estimation des paramètres pourra être précise ; or ces résultats montrent qu'il est préférable de créer un plan contenant un nombre pair d'expériences pour ce problème. L'ajout d'une nouvelle expérience à un tel plan a pour conséquence de diminuer la valeur du déterminant normé. Pour ce problème, un plan contenant un nombre impair d'expériences est sous-optimal par rapport à un plan contenant un nombre pair d'expériences.

Ce résultat élémentaire illustre également, sous les hypothèses que l'on a imposées sur le bruit, la relation entre la variance des paramètres et le placement des points sur le domaine d'étude. Ce lien est aussi décrit à l'aide de la figure 4.5.

Sur cette figure, nous pouvons voir 4 points **A**, **B**, **A'** et **B'**. Pour chacun d'eux, nous avons la même incertitude symbolisée par la double flèche. Les deux points **A** et **B** donnent, dans le pire cas, des paramètres estimés de la droite très éloignés des paramètres recherchés, alors que les points **A'** et **B'** donnent, même dans le pire cas, des paramètres estimés proches de ceux recherchés. On voit donc très nettement l'influence de la répartition des points sur la variance des paramètres.

Le placement des points au bord du domaine est justifié par l'existence d'un bruit expérimental sur les données. On peut remarquer, néanmoins, que dans le cadre déterministe, la planification au bord du domaine ne se justifie plus. Les plans que nous avons décrits sont optimaux par rapport à la minimisation de la région de confiance des paramètres dépendante du bruit expérimental. En l'absence de ce bruit, et pour l'exemple de la figure 4.5 le plan qui consiste à réaliser les expériences en **A** et **B** est tout aussi valable.

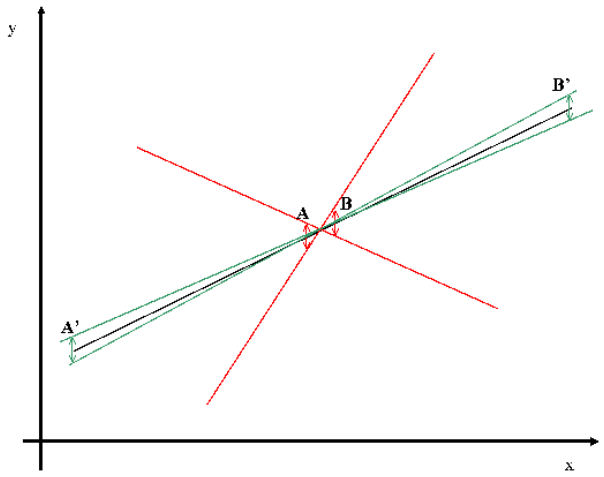



FIG. 4.5 – Lien entre placement des points et variance des paramètres

### 4.4.3 Les leviers et la D-optimalité

Reprenons l'exemple précédent du modèle linéaire à un facteur $y = \theta_0 + \theta_1 x$. Spécifions un plan D-Optimal. Dans le cas où le nombre d'expériences $N$ est pair, la matrice d'information est $(\mathbf{X}'\mathbf{X}) = N\mathbf{I}$. Le levier de l'exemple $x_i$ est donc égal à $2/N$ quel que soit $i$ puisque $x_i = +/-1$. Dans le cas $N$ pair, la D-optimalité correspond à l'uniformisation des leviers.

Dans le cas $N$ impair, la matrice d'information est égale à :

$$(X'X) = \frac{1}{N^2-1}\begin{pmatrix} N & +/-1 \\ +/-1 & N \end{pmatrix}$$

Les leviers $h_{ii}$ sont :

$$\begin{aligned} h_{ii} &= \frac{2(N+/-1)}{N^2-1} \\ &= \frac{2}{N+/-1} \end{aligned}$$

Dans ce cas, l'uniformisation exacte des leviers n'est pas obtenue par la D-optimalité.

## 4.5 Les plans optimaux pour modèles non linéaire par rapport aux paramètres

Nous allons décrire dans ce paragraphe, la construction de plans d'expériences optimaux pour modèles non linéaires. Pour une description plus détaillée voir [GAUCHI 97B].

### 4.5.1 La D-optimalité locale

Le principe de l'optimalité locale est dû à Chernoff [CHERNOFF 53]. On procède alors à un développement de Taylor au premier ordre du modèle autour du jeu de paramètres $\boldsymbol{\theta}_{\mathcal{L}}$ pour

lequel la fonction de coût des moindres carrés est minimum.

$$f(\mathbf{x}, \boldsymbol{\theta}) \simeq f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}}) + \mathbf{Z}(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}})(\boldsymbol{\theta} - \boldsymbol{\theta}_{\mathcal{L}})$$

où $\mathbf{Z}$ est la matrice jacobienne du modèle définie par :

$$[\mathbf{Z}(\mathbf{x}_i, \boldsymbol{\theta})]_{ij} = \Big(\frac{\partial f(\mathbf{x}, \boldsymbol{\theta})}{\partial \theta_j}\Big)_{\mathbf{x}=\mathbf{x_i}}$$

$\mathbf{Z}$ est une matrice $(N, p)$ où $N$ est le nombre d'exemples du plan d'expériences et $p$ le nombre de paramètres du modèle. L'emploi de la matrice jacobienne permet de faire une approximation locale du modèle, approximation qui est linéaire par rapport aux paramètres et linéaire en les dérivées partielles du modèle par rapport aux paramètres. La matrice jacobienne du modèle joue alors le même rôle que la matrice du modèle $\mathbf{X}$ dans le cadre linéaire[16]. Les définitions des critères d'optimalité des paragraphes précédents restent valables "localement". Le critère de D-optimalité devient le critère de D-optimalité locale que nous noterons D($\boldsymbol{\theta}_{\mathcal{L}}$)-optimalité et souvent désigné par D($\boldsymbol{\theta}_0$)-optimalité dans la littérature où $\boldsymbol{\theta}_0$ désigne une valeur *a priori* supposée représenter la vraie valeur. La D-optimalité locale a été étudiée par Vila [Vila 91], MacKay [MacKay 92a], Cohn [Cohn 94] et plus récemment par Issanchou et Gauchi [Issanchou & Gauchi 04] et Witczak [Witczak 06].

D'un point de vue algorithmique, on utilise les mêmes algorithmes que dans le cas linéaire en remplaçant la matrice $\mathbf{X}$ par la matrice $\mathbf{Z}$. Cependant, la matrice $\mathbf{Z}$ ne peut être obtenue que si un premier jeu de paramètres a été estimé. Donc, avant de pouvoir spécifier des expériences d'optimalité locale, il faut un premier plan d'expériences afin de créer un premier modèle. Dans la suite de ce mémoire, ce premier plan d'expériences sera créer par la méthode d'échantillonnage LHS.

Il existe également des plans d'expériences optimaux pour modèles non linéaires qui n'utilisent pas la linéarisation du modèle non linéaire autour de $\boldsymbol{\theta}_{\mathcal{L}}$ ; une de ces familles est celle fondée sur le critère de X-optimalité.

### 4.5.2 La X-optimalité

Dans cette approche [Vila 85], [Vila 86], [Vila & Gauchi 07] on ne fonde plus le critère d'optimalité sur la linéarisation du modèle autour de $\boldsymbol{\theta}_{\mathcal{L}}$, mais sur la minimisation de l'espérance du volume d'une région de confiance exacte pour le paramètre $\boldsymbol{\theta}$. Pour une description détaillée de ce critère voir [Gauchi 97b]. Cependant, ce critère est très lourd numériquement et il n'est pas exploitable pour des modèles contenant un très grand nombre de paramètres comme les réseaux de neurones. De plus, l'existence d'un bruit expérimental est indispensable pour créer de tels plans d'expériences. C'est pourquoi, dans la suite du mémoire, nous comparerons la méthode d'apprentissage actif que nous proposons à des plans de D-optimalité locale pour lesquels la variance de bruit de mesure tend vers zéro.

[16] Si le modèle est linéaire par rapport aux paramètres les matrice $\mathbf{X}$ et $\mathbf{Z}$ sont identiques.

## 4.6 Conclusion

Ce chapitre a été consacré à la présentation des critères d'optimalité les plus utilisés. Nous avons décrit l'algorithme d'échange double de Fedorov, qui permet de construire des plans optimaux pour modèles linéaires par rapport aux paramètres à partir de ces critères d'optimalité. Nous avons vu, ensuite, qu'en remplaçant la matrice du modèle $\mathbf{X}$ par la matrice jacobienne $\mathbf{Z}$ du modèle non linéaire pour la valeur de paramètre $\boldsymbol{\theta}_{\mathcal{L}}$, nous pouvions obtenir des plans d'expériences pour modèles non linéaire en appliquant les mêmes règles que dans le cadre linéaire : il s'agit donc d'une optimalité locale. Dans le dernier chapitre, nous comparerons la D-optimalité locale à la méthode LDR que nous proposons.

# Chapitre 5

# Apprentissage actif

**Sommaire**


**5.1 Introduction . . . . . 137**
**5.2 Principe général . . . . . 137**
**5.3 La méthode Learner Disagreement by experiment Resampling . . . 139**
5.3.1 Le Bagging . . . . . 139
5.3.2 Choix de l'exemple à replanifier . . . . . 140
5.3.3 Un exemple didactique simple . . . . . 142
**5.4 Comparaison entre méthodes pour la modélisation du processus de Friedman . . . . . 145**
5.4.1 La fonction de Friedman . . . . . 145
5.4.2 Procédure de test et résultats . . . . . 147
**5.5 Comparaison entre méthodes pour le processus de Homma et Saltelli 149**
5.5.1 La fonction de Homma et Saltelli . . . . . 149
5.5.2 Choix de l'architecture optimale pour le modèle de Homma et Saltelli . 151
5.5.3 Procédure de test et résultats . . . . . 151
5.5.4 Zones de replanification . . . . . 154
5.5.5 Temps de calcul . . . . . 156
5.5.6 Évolution de l'erreur de généralisation en fonction du nombre d'exemples 156
**5.6 Conclusion . . . . . 159**

## 5.1 Introduction

Nous allons introduire, dans ce chapitre, le principe général de l'apprentissage actif ("active learning") avant d'exposer la méthode LDR (Learner Disagreement from experiment Resampling) que nous proposons [Gazut & Martinez 05], [Gazut *et al.* 06] et [Gazut *et al.* 07]. Comme son nom l'indique, la méthode LDR utilise le ré-échantillonnage des exemples de la base d'apprentissage pour spécifier de nouveaux points. De nombreuses méthodes de ré-échantillonnage peuvent être utilisées ; nous utiliserons dans ce chapitre le bagging (notre méthode de planification sera alors appelée LDR-bagging), que nous décrirons, et le leave-one-out (planification par LDR-leave-one-out). Si la méthode de ré-échantillonnage n'est pas spécifée, c'est que nous utilisons le bagging. Nous illustrerons d'abord la méthode LDR sur des exemples "jouets" simples puis, nous testerons la méthode sur des cas tests plus complexes (dus à *Friedman* et à *Homma & Saltelli*), en comparant les résultats avec ceux obtenus avec des bases d'exemples obtenues à l'aide de la D-optimalité locale, et avec des bases d'exemples obtenues de manière aléatoire.

## 5.2 Principe général

La figure 1.1 du premier chapitre, présente un schéma de l'apprentissage supervisé avec un générateur d'exemples, un superviseur (la fonction à approcher) et un apprenti (le modèle à concevoir). Cet apprentissage peut être qualifié de passif : l'apprenti ne recoit que les données délivrées par le générateur. En apprentissage actif, l'apprenti lui-même, à l'aide de procédures que nous décrirons, a la capacité de selectionner, parmi un grand nombre d'expériences réalisables (souvent un maillage de l'espace des entrées), celles qui doivent être ajoutées aux données initiales [Cohn *et al.* 94] et [Melville & Mooney 04] (figure 5.1).

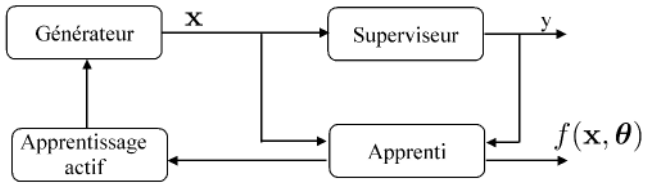



Fig. 5.1 – Schéma de l'apprentissage actif.

Les techniques d'apprentissage actif peuvent être résumées par 3 étapes :

- Entraîner l'apprenti à partir de la base d'apprentissage courante
- Choisir l'expérience $\mathbf{x}$ à réaliser parmi les expériences candidates. Cet ensemble d'expériences candidates est généralement représenté, dans le domaine d'étude, par un maillage obtenu par discrétisation de celui-ci.
- Réaliser l'expérience correspondant au point $\mathbf{x}$ et ajouter le point $(\mathbf{x}, y(\mathbf{x}))$ à la base d'apprentissage.

On itère la procédure autant de fois que l'on souhaite de nouveaux points.

L'apprentissage actif recouvre donc un ensemble de méthodes visant à sélectionner les exemples de façon à améliorer la construction des modèles. Cet objectif est aussi celui des plans d'expériences. Un lien existe donc entre les deux approches.

Dans [Cohn *et al.* 90] et [MacKay 92b] ce lien est établi notamment avec la D-optimalité. Les plans optimaux sont dus aux travaux de Kiefer [Kiefer 59] puis Fedorov [Fedorov 72]. En effet, leur approche est fondée sur le calcul de la matrice de dispersion, opérateur qui conditionne la précison de l'estimation. La matrice de dispersion est utilisée pour estimer le gain d'information apporté par une nouvelle requête au superviseur. Le gain d'information a posteriori, avant la connaissance de la réponse à la nouvelle requête, est estimé par les techniques de Bayes. La méthode nécessite donc plusieurs hypothèses sur les distributions a priori et les distributions conditionnelles. Pour les modèles non linéaires par rapport aux paramètres, la matrice de dispersion est obtenue par linéarisation.

En apprentissage actif, l'approche est différente. Elle est fondée sur la notion de comité d'experts (Query by Committee) introduite par [Seung *et al.* 92]. Chaque expert est un modèle hypothèse différent. L'ensemble des hypothèses est utilisé pour établir une mesure de la différence entre les prédictions appelée divergence des prédictions. La nouvelle requête au superviseur est celle qui maximise la divergence des prédictions. Les méthodes de comité d'experts, se distinguent principalement par la façon de construire les experts et par la méthode de calcul de la divergence des prédictions.

Dans [Seung *et al.* 92], en classification supervisée, les modèles hypothèses sont construits par un algorithme stochastique fournissant un modèle différent à chaque apprentissage. Dans [Gilardi & Faraj 04] en régression non linéaire par réseaux de neurones, c'est l'initialisation aléatoire des paramètres qui permet de produire des modèles différents. Cette méthode ne peut pas être utilisée dans les approches classiques linéaires des plans d'expériences, où le modèle est obtenu par un algorithme d'apprentissage déterministe. Cette approche de construction d'hypothèses ne nous paraît donc pas vraiment justifiée pour notre problème dans le sens où nous mettons en place une méthode indépendante du choix de la famille de fonctions utilisée.

La seconde méthode pour construire les différentes hypothèses est fondée sur les techniques de ré-échantillonnage par bootstrap par utilisation du Bagging [Breiman 96] ou du Boosting [Shapire 99]. Ces approches ont été proposées et développées principalement en classification supervisée avec des mesures d'utilité sur la requête fondée sur les notions d'entropie [Melville & Mooney 04] et [Abe & Mamitsuka 98].

Notre approche en apprentissage actif est similaire aux méthodes de constructions des hypothèses fondées sur le bootstrap, et donc similaire à celle du Bagging. Elle est développée dans le cadre de la régression qui a été beaucoup moins étudiée que la classification supervisée. Nous avons utilisé la variance des prédictions comme mesure de divergence des prédictions. Les réseaux de neurones utilisés comme hypothèses ont été construits par apprentissage, avec les mêmes valeurs initiales de paramètres, sur des bases obtenus par ré-écchantillonnage bootstrap. La variabilité des modèles obtenus représente donc la variabilité de l'apprentissage par rapport au procédé d'échantillonnage utilisé pour engendrer la base d'exemples. Notre méthode se distingue donc de celles qui utilisent une approximation de la matrice de dispersion (inverse de la

matrice d'information des plans d'expériences). Une comparaison sera faite avec la D-Optimalité.

## 5.3 La méthode Learner Disagreement by experiment Resampling

Comme nous l'avons décrit dans les chapitres précédents, $\boldsymbol{\theta}_{\mathcal{L}}$ désigne le vecteur de paramètres du modèle obtenu à partir de la base d'apprentissage $\mathcal{L}$ et pour lequel la fonction de coût est minimum. La variabilité de ce vecteur de paramètres par rapport à la base d'apprentissage $\mathcal{L}$ a été étudiée dans le cadre du Bagging que nous allons décrire, et que nous mettrons en oeuvre dans le cadre de la méthode LDR que nous proposons.

### 5.3.1 Le Bagging

Le *Bagging* (acronyme de **B**ootstrap **agg**regat**ing**) a été proposé par Breiman [Breiman 96]. Supposons les points de l'échantillon $\mathcal{L}$ choisis par un tirage aléatoire selon une densité de probabilité $P(\mathbf{x})$ définie dans $U$. En notant $E_{\mathcal{L}}$ l'espérance calculée sur l'ensemble des échantillons possibles $\mathcal{L}$ de même taille, le modèle agrégé est défini de la manière suivante :

$$f_A(\mathbf{x}) = E_{\mathcal{L}}[f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}})] \tag{5.1}$$

Parmi l'ensemble des modèles construits sur des échantillons de même taille $\mathcal{L}$, le modèle agrégé est celui qui minimise l'espérance de $\|f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}}) - y(\mathbf{x})\|^2$ :

$$\|f_A(\mathbf{x}) - y(\mathbf{x})\|^2 = \|E_{\mathcal{L}}[f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}})] - y(\mathbf{x})\|^2 \leq E_{\mathcal{L}}[\|f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}}) - y(\mathbf{x})\|^2]$$

La solution optimale peut être estimée par la moyenne des modèles sur une famille de bases $\mathcal{L}^{*i}$ et ne peut évidemment pas être calculée à partir d'une seule base d'apprentissage. Obtenir une famille de bases $\mathcal{L}^{*i}$ peut être couteux ; le Bagging consiste alors à approcher l'espérance $f_A(\mathbf{x})$ par ré-échantillonnage bootstrap de la base initiale. On n'utilise, ainsi, qu'une unique base $\mathcal{L}$ à partir de laquelle on crée une famille de bases $\mathcal{L}^{*i}$ constituée des répliques bootstrap de $\mathcal{L}$.

En notant $\mathcal{L}_N$ la base d'apprentissage initiale constituée de $N$ exemples et $\mathcal{L}_N^{*b}$ la réplique $b$ de $N$ exemples obtenue à partir de $\mathcal{L}_N$, l'estimation de l'espérance, obtenue par bootstrap à partir de $B$ répliques est :

$$f_{N,\mathcal{B}}(\mathbf{x}) = \frac{1}{B} \sum_{b=1}^{B} f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}_N^{*b}})$$

La figure 5.2 est un schéma de la méthode bagging.

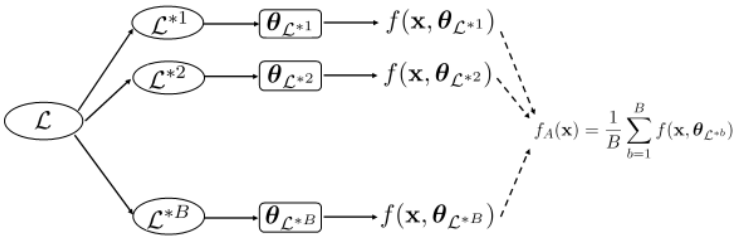



FIG. 5.2 – Modèle agrégé obtenu à partir du ré-échantillonnage bootstrap.

La méthode bagging améliore l'efficacité de la prediction si une perturbation de la base d'apprentissage entraine des changements significatifs sur le modèle construit.

### 5.3.2 Choix de l'exemple à replanifier

De même que l'on estime l'espérance des valeurs de prédiction, nous pouvons estimer leur variance en utilisant le bootstrap. L'expression de l'estimation de la variance est la suivante :

$$s^2_{\mathcal{B}}[f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}_N})] = \frac{1}{B}\sum_{b=1}^{B}(f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}^{*b}_N}) - f_{N,\mathcal{B}}(\mathbf{x}))^2$$

$s^2_{\mathcal{B}}[f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}_N})]$ représente donc la variance des sorties estimées par les modèles, dont les bases d'apprentissage sont des répliques bootstrap, autour du modèle agrégé $f_A(\mathbf{x})$. La méthode d'apprentissage actif que nous proposons consiste à ajouter de nouvelles expériences dans les régions de l'espace des entrées où l'estimation bootstrap de la variance des prédictions est la plus grande, c'est-à-dire dans les zones de l'espace des entrées où les modèles construits à partir des répliques bootstrap sont le plus en désaccord. Notre méthode s'inscrit dans le cadre des méthodes QbC (Query by Committes). En reprenant la métaphore du maître et de l'élève de l'apprentissage statistique, dans le cadre de notre méthode, il s'agit d'un maître et d'une classe d'élèves. Chaque élève apprend à partir d'une partie des données (répliques bootstrap), le maître pose des questions (les points du maillage), les élèves donnent leurs réponses (l'estimation aux points de maillage) et demandent au maître la réponse exacte à la question qui a provoqué le plus grand désaccord.

La méthode que nous proposons peut se résumer de la manière suivante :

– Trouver le point pour lequel la variance des prédictions est maximale

$$\mathbf{x}_{new} = ArgMax_{\mathbf{x}\in U} s^2_{\mathcal{B}}[f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}_N})]$$

– Réaliser l'expérience numérique correspondant à l'entrée $\mathbf{x}_{new}$ ($y(\mathbf{x}_{new})$) et ajouter l'expérience $(\mathbf{x}_{new}, y(\mathbf{x}_{new}))$ à la base d'apprentissage initiale $\mathcal{L}_N$ pour obtenir la base $\mathcal{L}_{N+1}$.

$$\mathcal{L}_{N+1} = \mathcal{L}_N \cup \{(\mathbf{x}_{new}, y(\mathbf{x}_{new}))\}$$

Cet apprentissage actif peut être interrompu lorsque la décroissance de la variance de prédiction n'est plus significative ou lorsque le nombre maximum d'expériences défini a priori est atteint. L'algorithme LDR ci-dessous résume la méthode.

---

**L'algorithme LDR**

---

**En se donnant :**
$T$ - la base d'apprentissage
$P$ - le maillage de l'espace des entrées
$y$ - le processus inconnu que l'on cherche à modéliser
$k$ - le nombre d'expériences que l'on souhaite ajouter
$N$ - la taille initiale de la base d'apprentissage
Répéter $k$ fois :

1. Créer $B$ répliques bootstrap $\mathcal{L}_N^{*b}$
2. Calculer le modèle agrégé par bootstrap $f_{N,\mathcal{B}}(\boldsymbol{x})$
3. $\forall x_j \in P$ calculer l'estimation de la variance de prédiction $s^2_{\mathcal{B}}[f(\boldsymbol{x}, \boldsymbol{\theta}_{\mathcal{L}_N})]$
4. Déterminer le point de $P$ pour lequel la variance de prédiction est maximale, noté $\boldsymbol{x}_{new}$
5. Extraire $\boldsymbol{x}_{new}$ de $P$ and ajouter $(\boldsymbol{x}_{new}, y(\boldsymbol{x}_{new}))$ à $T$ ; $(N \leftarrow N + 1)$.

---

La méthode reste inchangée si l'on utilise d'autres méthodes de ré-échantillonnage que le bootstrap. Par exemple, pour des modèles linéaires par rapport aux paramètres comme les polynômes, on peut utiliser le leave-one-out pour estimer la variance des prédictions. Le calcul de cette variance est alors beaucoup plus simple, plus immédiat et ne necessite pas la phase de choix de modèles. En effet, un seul apprentissage et un seul calcul de la matrice chapeau $\mathbf{H}$ permettent de calculer cette variance, par l'intermédiaire du leave-one-out virtuel vu au chapitre 2. La variance des paramètres obtenue par leave-one-out s'écrit :

$$s^2_{loo}(\boldsymbol{\theta}) = \mathbf{X}^+ \boldsymbol{\Lambda} (\mathbf{X}^+)'$$

où $\mathbf{X}^+ = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$ est la pseudo inverse de $\mathbf{X}$, et $\boldsymbol{\Lambda}$ est la matrice d'élement $\Lambda_{ij} = \frac{1}{N}\alpha_i^2\delta_{ij} - \frac{1}{N^2}\alpha_i\alpha_j$ avec $\alpha_i = r_i/(1-h_{ii})$.

La variance des prédictions au point $\mathbf{x}$ s'exprime comme :

$$s^2_{loo}(f(\mathbf{x}, \boldsymbol{\theta})) = \phi(\mathbf{x})'\mathbf{X}^+\boldsymbol{\Lambda}\mathbf{X}^{+'}\phi(\mathbf{x})$$

Le point que l'on replanifie est, parmi un ensemble de points candidats, celui pour lequel la variance est maximale :

$$\mathbf{x}_{new} = ArgMax_{\mathbf{x}\in U}||\phi(\mathbf{x})'\mathbf{X}^{+}\mathbf{\Lambda}^{1/2}||$$

Nous allons aborder la planification d'expériences pour modèles linéaires par rapport aux paramètres à la fin de ce chapitre au paragraphe 5.5.6 et comparer la qualité des bases replanifiées dans ce cas à la qualité des bases LHS.

### 5.3.3 Un exemple didactique simple

#### Le sinus cardinal en dimension 1

Pour illustrer simplement la méthode, nous avons utilisé, comme processus générateur des données, la fonction sinus cardinal en dimension 1 sur le domaine $[-4; 10]$. La base d'apprentissage initiale contient dix exemples non uniformement distribués dans l'espace des entrées. Nous estimons la variance des prédictions par Bootstrap sur l'ensemble du domaine et nous ajoutons, dans la base d'exemples initiale, le point $(\mathbf{x}, y(\mathbf{x}))$ où $\mathbf{x}$ est le point de maillage pour lequel la variance des prédictions est maximale et $y$ la valeur de la réponse du processus à modéliser en ce point. Les modèles créés à partir des répliques bootstrap de la base d'apprentissage sont des réseaux de neurones contenant 4 neurones cachés.

La figure 5.3 présente les résultats pour les six premiers points ajoutés aux exemples d'apprentissage initiaux.

La variance des prédictions décroit de manière significative à chaque itération au voisinage des nouveaux points expérimentaux, et décroit de manière globale sur l'ensemble du domaine. Nous aurions pu interrompre la procédure à la cinquième étape car la variance des prédictions n'y est plus significative et il n'y a pas de réel désaccord entre les modèles.

La figure 5.4 présente l'évolution de l'erreur de généralisation, estimée à partir d'un maillage très fin du domaine d'étude (cf equation (3.10)), en fonction du nombre d'expériences ajoutées. Cette erreur de généralisation a été calculée pour des modèles ayant appris sur les bases d'exemples créées par la méthode LDR d'une part, et par une planification aléatoire d'autre part.

Cette courbe n'est pas statistiquement significative, puisque nous avons comparé une base de données obtenue par LDR à une seule réalisation aléatoire d'une base ; elle est présentée ici de manière uniquement illustrative. Une étude plus significative sera décrite dans la section 5.4 de ce chapitre.

#### Le sinus cardinal en dimension 2

Nous avons utilisé également le sinus cardinal en dimension 2 ($\frac{sin(\sqrt{x^2+y^2})}{\sqrt{x^2+y^2}}$) en utilisant cette fois-ci comme base d'apprentissage initiale un tirage LHS de 50 points sur le domaine $[-4; 10]^2$. Les modèles construits à partir des répliques bootstrap de la base d'apprentissage sont des

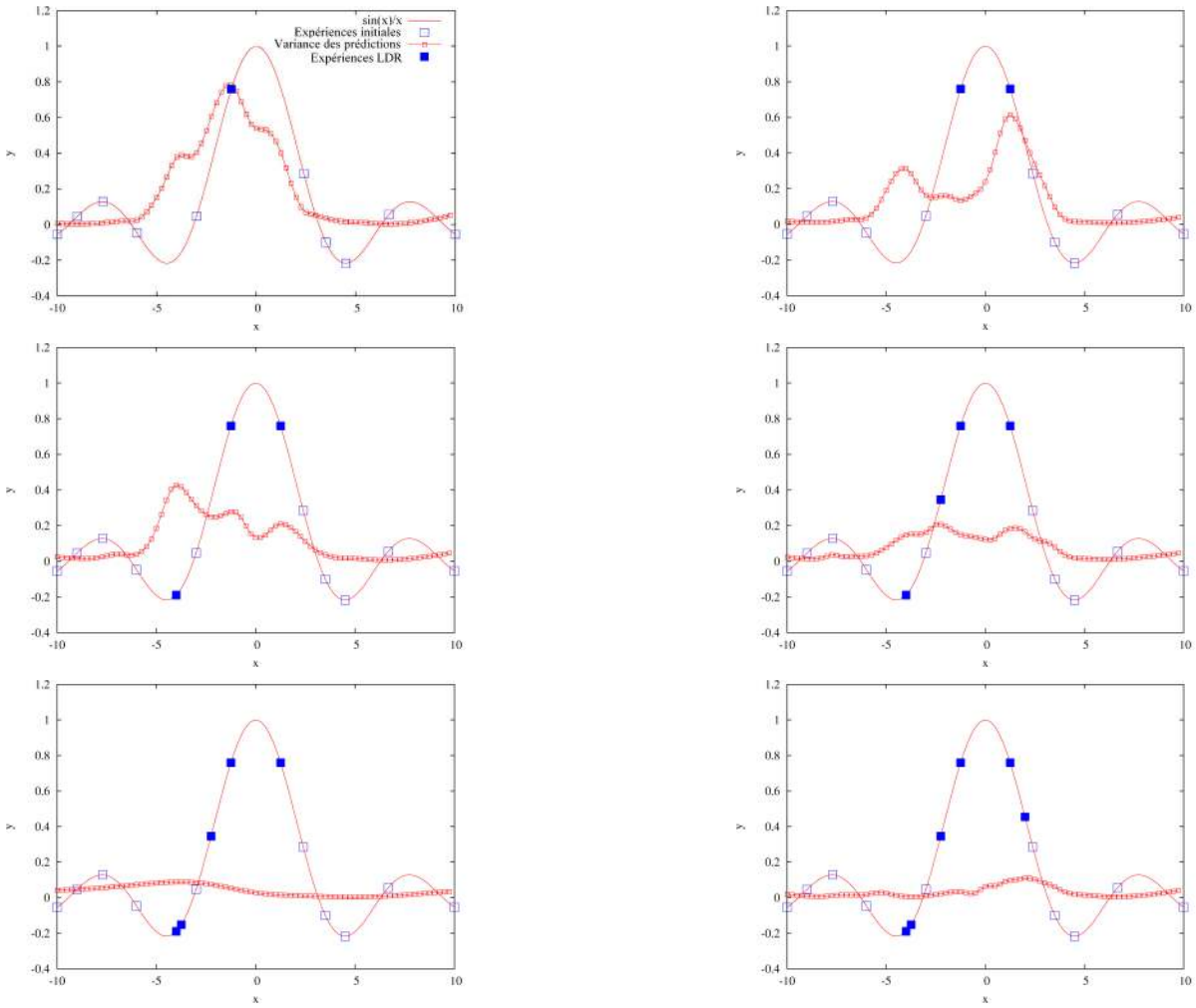



FIG. 5.3 – Apprentissage actif par la méthode LDR sur le sinus cardinal

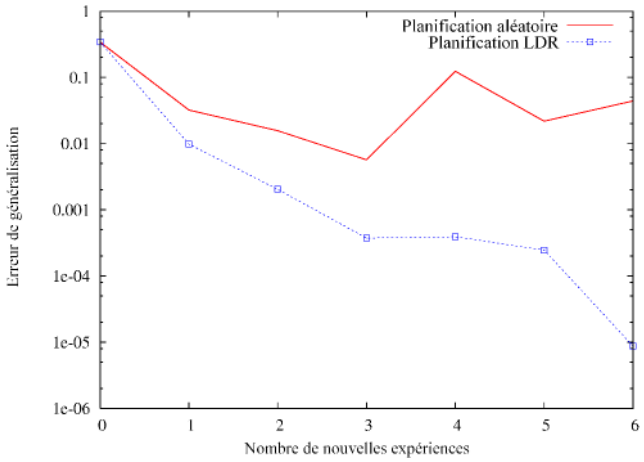



FIG. 5.4 – Comparaison de l'erreur de généralisation entre des modèles ayant appris sur les bases engendrée par LDR d'une part, et par planification aléatoire d'autre part.

réseaux de neurones de type MLP avec 6 neurones cachés. La figure 5.5 présente la répartition des points de la base d'apprentissage initiale sur le domaine d'étude.

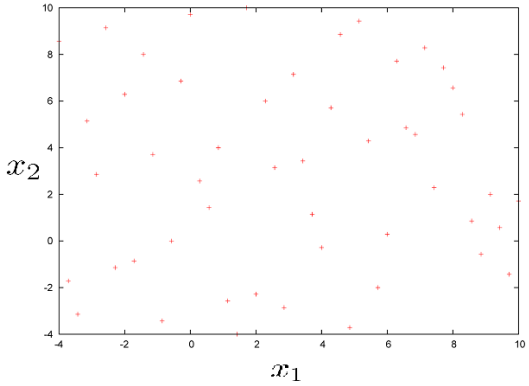



FIG. 5.5 – Répartitions des points LHS sur le domaine $[-4;10]^2$ pour le sinus cardinal en dimension 2.

La figure 5.6 présente ces mêmes points répartis sur la surface sinus cardinal. Le voisinage de l'origine, où la fonction est maximale, ne comporte pas plus de points que les régions où la fonction varie moins.

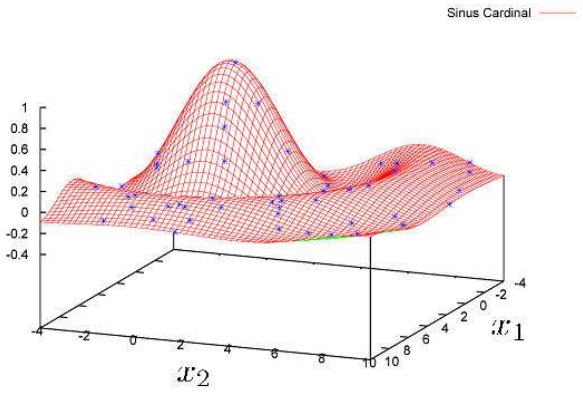



FIG. 5.6 – Répartitions des points LHS sur la surface sinus cardinal.

Nous avons utilisé la même démarche que précédemment. La figure 5.7 présente le profil de variance des prédictions à l'itération 0.

Le maximum de variance est atteint au voisinage du point $(-4;1)$ ; cette zone correspond à une carrence d'information (il y a très peu de points au bord entre les points $(-4,-2)$ et $(-4,8)$). Comme precedemment, de nouveaux exemples sont ajoutés à la base d'apprentissage aux points où la variance de prédiction est maximale. La figure 5.8 représente les nouveaux profils de variance à chaque itération. Le numéro de l'itération correspondant à chacun des profils de variance est noté en haut à droite de chaque figure.

La figure 5.9 montre les points ajoutés à la base d'apprentissage.

Un certain nombre de points sont ajoutés aux bords du domaine ; les autres points ajoutés

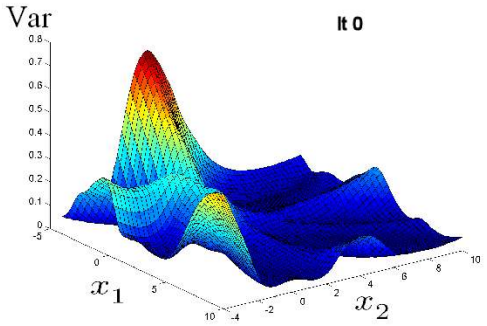



FIG. 5.7 – Profil de variance des prédictions estimée par Bootstrap pour le sinus cardinal en dimension 2.

sont concentrés autour de l'origine, c'est-à-dire dans la zone de forte variations de la sortie du processus sinus cardinal.

## 5.4 Comparaison entre méthodes pour la modélisation du processus de Friedman

Nous allons étudier à présent des processus générateurs plus complexes que les précédents. Nous considérerons deux exemples classiques, celui de Friedman et celui de Homma et Saltelli. Pour ces deux processus, nous postulons des modèles non linéaires de type MLP que nous avons décrits au premier chapitre. L'apprentissage actif LDR, la D-optimalité locale appliquée aux réseaux MLP, et la planification aléatoire seront mis en oeuvre pour constituer des bases d'apprentissage. Les capacités de généralisation des modèles construits à partir de ces bases seront comparées.

### 5.4.1 La fonction de Friedman

Le processus générateur des données est la fonction de Friedman :

$$y(\mathbf{x}) = \theta_1 sin(\pi x_1 x_2) + \theta_2 (x_3 - \theta_3)^2 + \theta_4 x_4 + \theta_5 x_5 \tag{5.2}$$

avec $\theta_1 = \theta_4 = 0.4$, $\theta_2 = 0.8$, $\theta_3 = 0.5$, $\theta_5 = 0.2$ et $x_i \in [0;1] \forall i = 1, ..., 5$

Cette fonction de $\mathcal{R}^5$ dans $\mathcal{R}$ joue le rôle d'un code de calcul : aucun bruit de mesure n'est ajouté aux résultats numériques. Les trois méthodes de planification que nous allons tester utilisent le même plan d'expériences initial obtenu par un tirage LHS de 100 expériences. Les modèles neuronaux que nous utilisons sont des réseaux MLP avec 6 neurones cachés[17] à fonctions d'activation logistiques et un neurone de sortie linéaire. 30 expériences supplémentaires seront planifiées.

[17] Une étude préalable à montré que cette architecture offrait un bon compromis entre biais et variance.

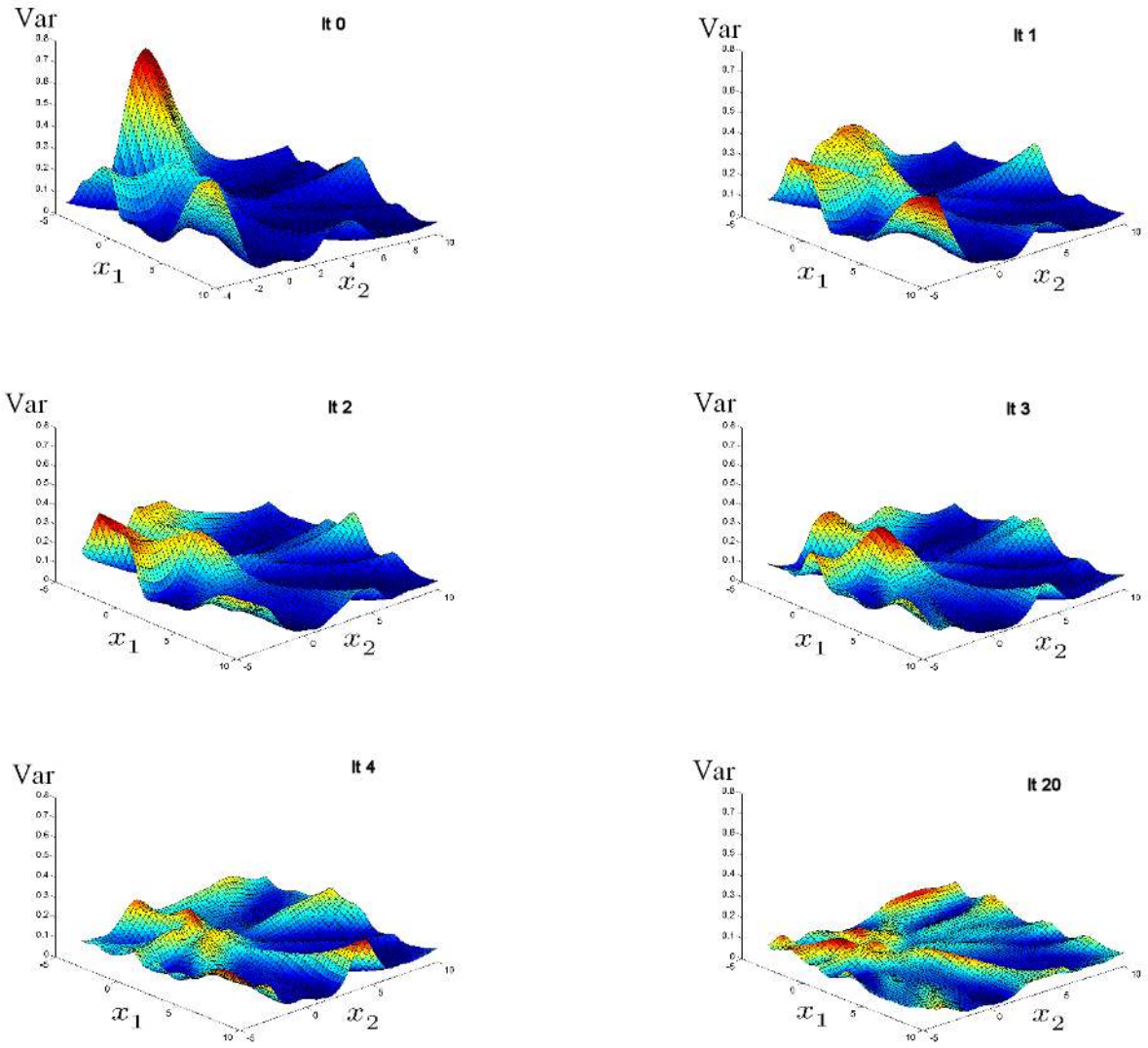



FIG. 5.8 – Profils de variance pour différentes itérations de replanification pour le sinus cardinal en dimension 2.

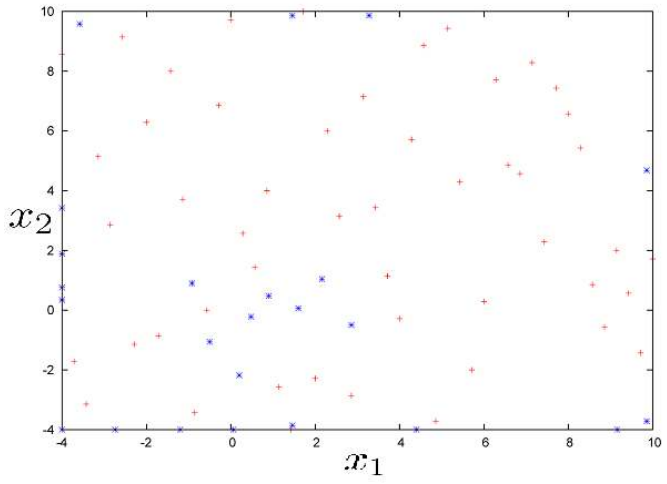



FIG. 5.9 – Répartition des points spécifiés par la méthode LDR parmi les points de la base d'apprentissage.

### 5.4.2 Procédure de test et résultats

Le premier objectif est l'estimation et la comparaison des capacités de généralisation de modèles dont l'apprentissage a été réalisé avec des bases obtenues par les méthodes D-optimales, LDR, et par tirage aléatoire. Nous étudierons ensuite l'évolution de l'erreur de généralisation en fonction du nombre de points ajoutés à la base initiale.

#### Comparaison entre plan D-optimal, plan LDR et plans aléatoires

L'utilisation d'un plan initial est indispensable pour la D-optimalité et la méthode LDR afin de procéder à la création des premiers modèles. Ils permettent de calculer la matrice jacobienne $\mathbf{Z}$ dans le cas de la D-optimalité, et de procéder au calcul de la variance des prédictions dans le cadre de la méthode LDR.

Les 30 expériences à ajouter aux 100 expériences du plan LHS initial seront spécifiées en une seule fois pour la D-optimalité, et point par point pour la méthode LDR. Pour les plans naïfs (aléatoires), 30 expériences seront ajoutées au hasard. Les plans obtenus par la D-optimalité et par la méthode LDR seront comparés entre eux, et à un ensemble de 1000 bases aléatoires différentes pour que les résultats soient statistiquement significatifs.

Un plan $P_1$ sera considéré meilleur qu'un plan $P_2$, pour un modèle postulé donné, si l'erreur de généralisation du modèle dont les paramètres ont été estimés à partir des exemples de $P_1$ est plus faible que l'erreur de généralisation du modèle dont les paramètres ont été estimés à partir des exemples de $P_2$. Cette erreur de généralisation est estimée à l'aide d'une base de test comprenant 20 000 exemples. Les modèles que nous postulons étant des réseaux de neurones, non linéaires par rapport aux paramètres, plusieurs estimations des paramètres, correspondant à des minima différents de la fonction de coût, peuvent être obtenues à partir d'un même ensemble d'apprentissage. Conformément aux résultats du paragraphe 3.5.2, nous retenons le modèle correspondant au plus faible minimum de la fonction de coût, parmi 30 valeurs initiales différentes des paramètres. Les composantes du vecteur des paramètres sont tirées aléatoirement dans l'in-

tervalle $[-\frac{1}{2}, \frac{1}{2}]$ correspondant à l'intervalle où la fonction d'activation des neurones cachés (la fonction logistique pour les réseaux de neurones que nous utilisons) est linéaire.

La figure 5.10 représente l'histogramme des 1000 erreurs de généralisation des modèles construits sur les bases aléatoires ainsi que l'erreur de généralisation des modèles dont l'apprentissage a été conduit sur le plan obtenu par la D-optimalité et le plan obtenu par la méthode LDR.

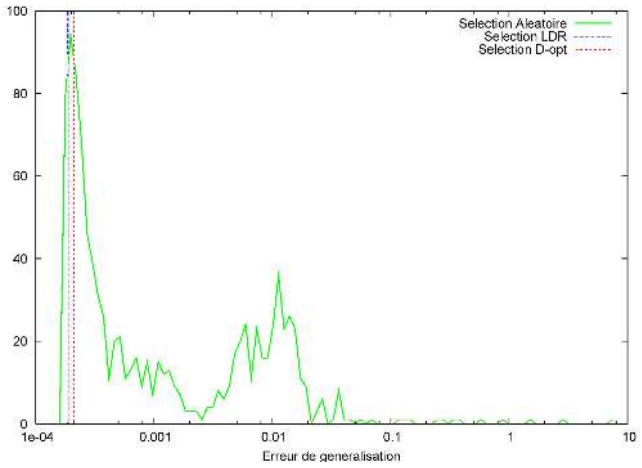



FIG. 5.10 – Histogramme des erreurs de généralisation de modèles dont les paramètres ont été estimés à partir de 1000 plans aléatoires. L'erreur de généralisation est évaluée sur un échantillon de 20 000 exemples. Les deux segments verticaux sont les erreurs de généralisation des modèles dont l'apprentissage a été conduit à partir des plans obtenus par la D-optimalité et la méthode LDR.

96,7 % des modèles obtenus à partir des bases aléatoires généralisent moins bien que celui obtenu à partir de la méthode LDR, 94,8 % d'entre eux généralisent moins bien que le modèle obtenu par apprentissage à partir du plan D-optimal. Ces deux méthodes de planification d'expériences sont plus efficaces que la planification aléatoire.

### Comparaison entre plan D-optimal et plan LDR

Grâce au modèle obtenu à partir de la base LHS initiale, nous avons fait une planification D-optimale et une planification LDR. Le test de comparaison de ces plans consiste à estimer les erreurs de généralisation de 1000 modèles dont les paramètres ont été initialisés différemment avant l'apprentissage.

La figure 5.11 représente les histogrammes des erreurs de généralisation de ces modèles. Nous avons créé 1000 modèles à partir de la base D-optimale et 1000 modèles à partir de la base LDR. Chacun de ces modèles est le meilleur modèle en apprentissage parmi 30 modèles initialisés différemment. La figure 5.12, représente la fonction de répartition de l'erreur de généralisation de ces modèles ; elle montre que les deux méthodes permettent d'obtenir des performances de généralisation très voisines, pour le processus générateur considéré et la complexité de modèles considérée.

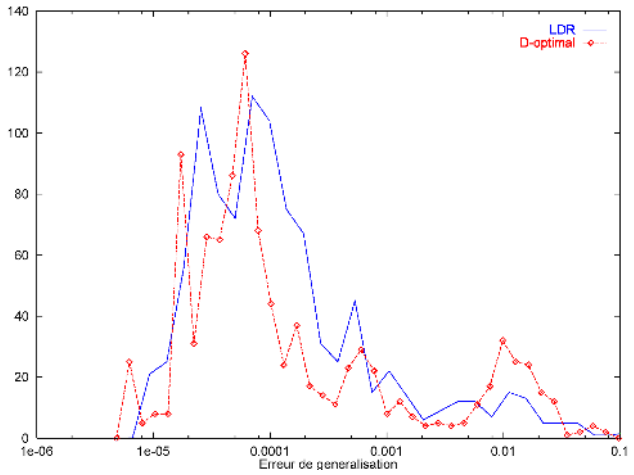



FIG. 5.11 – Histogramme des 1000 erreurs de généralisation pour la planification D-optimale et la planification LDR. Nous avons conduit l'apprentissage de 1000 modèles, chacun d'entre eux est le meilleur modèle en apprentissage parmi un ensemble de 30 modèles initialisés différemment. Nous avons calculé pour chacun d'eux l'erreur de généralisation sur un ensemble de test de 20 000 exemples. Cette figure représente l'histogramme des valeurs d'erreur de généralisation de ces 1000 modèles pour la D-optimalité et pour la méthode LDR.

#### Évolution de l'erreur de généralisation en fonction du nombre d'exemples

Nous étudions à présent l'évolution de l'erreur de généralisation en fonction du nombre de points ajoutés à la base initiale. Nous avons estimé l'erreur de généralisation des modèles qui apprennent sur les bases de $N$ exemples pour $N$ variant de 100 (base LHS initiale) à 130 (base finale). Pour chaque valeur de $N$, on applique la même règle de choix de modèle, qui consiste à prendre comme vecteur initial de paramètres, celui du meilleur modèle en apprentissage parmi 30 modèles entraînés.

La figure 5.13 représente l'évolution de l'erreur de généralisation en fonction du nombre de points ajoutés à la base initiale. L'erreur de généralisation décroît lorsque le nombre de points augmente, mais l'on n'observe pas de différences significatives entre les évolutions qui résultent des deux méthodes de planification.

## 5.5 Comparaison entre méthodes pour le processus de Homma et Saltelli

### 5.5.1 La fonction de Homma et Saltelli

Le processus générateur des données est la fonction de Homma et Saltelli :

$$y(\mathbf{x}) = sin(x_1) + 7sin^2(x_2) + 0.1x_3^4 sin(x_1) \tag{5.3}$$

avec $x_i \in [-\pi; \pi]\ \forall i = 1, ..., 3$

Cette fonction de $\mathcal{R}^3$ dans $\mathcal{R}$ joue le rôle d'un code de calcul : aucun bruit n'est ajouté aux données. Cette fonction est difficile à modéliser ; le paragraphe suivant est consacré au choix de

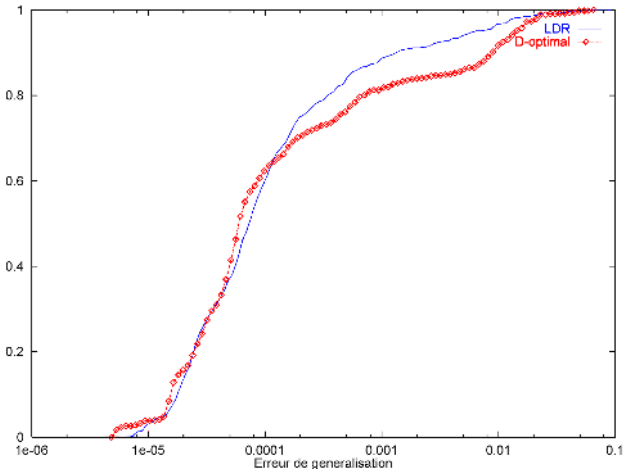



FIG. 5.12 – Fonction de répartition de l'erreur de généralisation pour la planification D-optimale et la planification LDR.

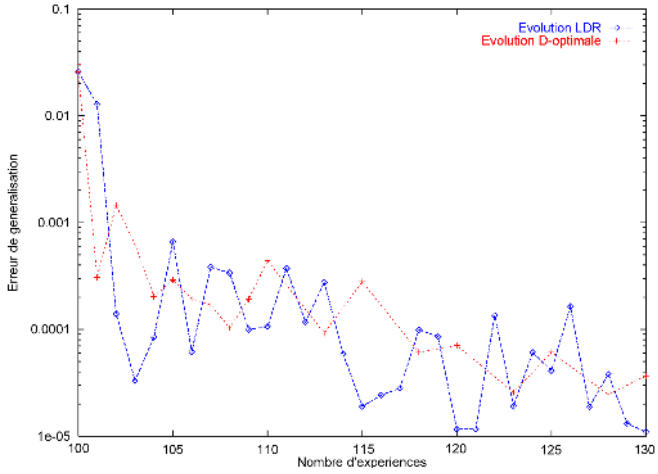



FIG. 5.13 – Évolution de l'erreur de généralisation des modèles dont l'apprentissage a été effectué sur les bases planifiées par la méthode LDR et la D-optimalité, en fonction du nombre d'exemples que contiennent ces bases.

l'architecture des réseaux de neurones qui permettent de modéliser les données obtenues.

### 5.5.2 Choix de l'architecture optimale pour le modèle de Homma et Saltelli

Afin de trouver l'architecture optimale des réseaux de neurones, nous avons calculé l'erreur quadratique moyenne d'apprentissage (EQMA) et l'erreur quadratique moyenne de test (EQMT) en fonction du nombre de neurones cachés. Pour chacune des architectures, nous avons reporté, sur la figure 5.14, les erreurs obtenues pour 20 modèles initialisés différemment. Nous avons également fait varier le nombre d'exemples $N$ de la base d'apprentissage initiale ($N = 50$, $N = 100$ et $N = 200$).

La première colonne de figures correspond à l'évolution de l'EQMA. La deuxième colonne correspond à l'évolution de l'EQMT. Les trois lignes de figures représentent respectivement les résultats pour $N = 50$, $N = 100$ et $N = 200$.

Les modèles construits à partir de la base d'apprentissage contenant 50 exemples n'ont pas de bonnes capacités de généralisation. Les modèles construits à partir de la base d'apprentissage à 200 exemples sont plus satisfaisants. On pourra difficilement observer, dans ce cas, l'impact des points ajoutés sur les capacités prédictives des modèles, car la base paraît suffisamment complète pour que de bons modèles puissent être créés. C'est pourquoi nous avons fait le choix d'utiliser une base d'apprentissage initiale contenant 100 exemples ainsi que des réseaux de neurones avec 12 neurones cachés. Nous pourrons ainsi observer l'apport des points planifiés sur les capacités de généralisation des modèles.

### 5.5.3 Procédure de test et résultats

Les tests doivent permettre d'estimer, comme précédemment, les performances des plans d'expériences obtenus à partir des méthodes D-optimal, LDR et aléatoire en termes d'erreur de généralisation des modèles construits à partir de ces bases.

#### Comparaison entre plan D-optimal, plan LDR et plans aléatoires

Pour les trois méthodes, un plan d'expériences initial obtenu par un tirage LHS permet de créer un premier modèle. Le plan LHS initial comporte 100 exemples, auxquels 60 expériences sont ajoutées.

La figure 5.15 représente les points du plan LHS initial en projection sur les plans $(Ox_1x_2)$, $(Ox_1x_3)$ et $(Ox_2x_3)$

Ce plan LHS initial permet d'explorer correctement le domaine d'étude.

Comme pour le cas du processus de Friedman, nous allons comparer les trois méthodes entre elles. Nous avons comparé les capacités de généralisation de modèles obtenus à partir du plan D-optimal et du plan créé par la méthode LDR aux capacités de généralisation de modèles obtenus par apprentissage à partir de 1000 bases aléatoires différentes. L'erreur de généralisation des modèles est estimée à l'aide d'une base de test comprenant 20 000 exemples. Nous utiliserons la même stratégie de choix de modèles que pour le cas précédent : le modèle sélectionné est celui qui a la plus petite erreur d'apprentissage parmi 30 modèles initialisés différements.

La figure 5.16 représente l'histogramme des 1000 modèles construits à partir des bases aléatoires, ainsi que l'erreur de généralisation des modèles dont l'apprentissage a été conduit sur les

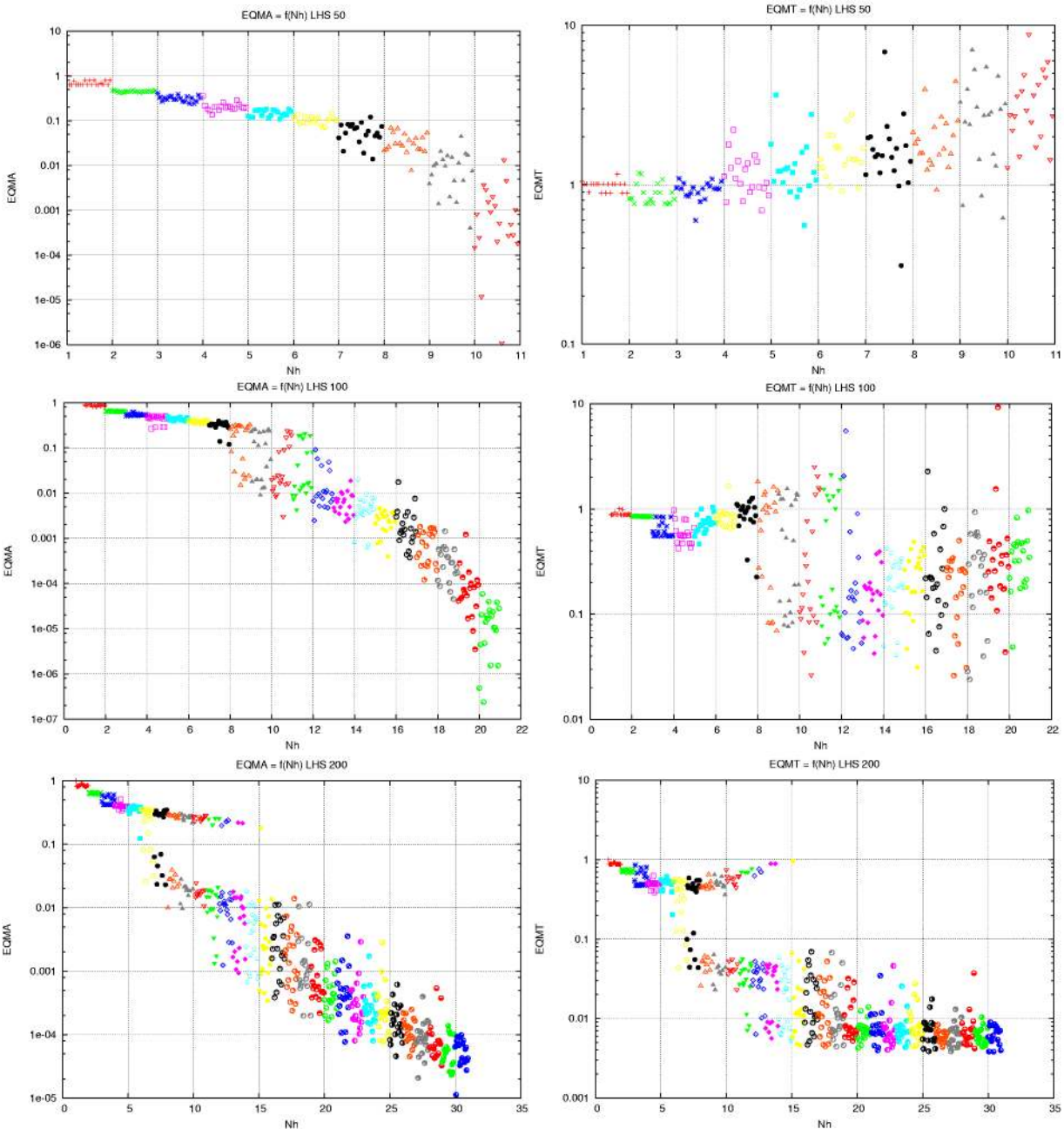



FIG. 5.14 – Erreur d'apprentissage et erreur de test pour différentes architectures et pour différentes tailles de bases d'apprentissage. Chaque architecture est représentée par un code de couleur différent.

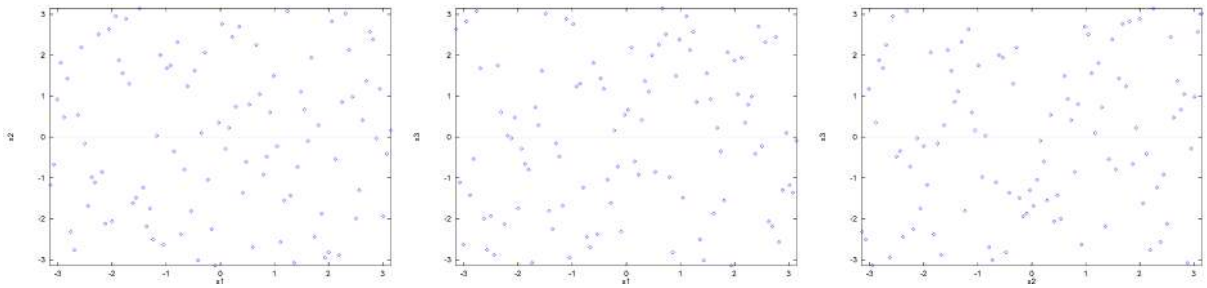


FIG. 5.15 – Plan LHS initial vu en projection.

plans obtenus par la D-optimalité et la méthode LDR.

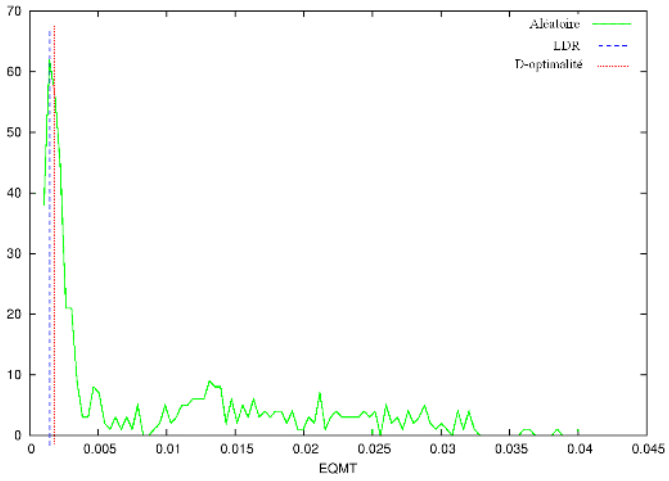



FIG. 5.16 – Histogramme des 1000 erreurs de généralisation des modèles obtenus par apprentissage à partir de bases obtenues par planification aléatoire. Nous avons conduit l'apprentissage de modèles (le meilleur modèle en apprentissage parmi 30) sur chacune des bases aléatoires. Les deux segments verticaux sont les erreurs de généralisation des modèles dont l'apprentissage a été conduit à partir des plans obtenus par la D-optimalité et la méthode LDR.

Comme attendu, la méthode LDR et la méthode D-optimale permettent d'obtenir de meilleurs résultats que la planification aléatoire. Les modèles construits à partir de la D-optimalité sont meilleurs que ceux obtenus par planification aléatoire dans 86 % des cas. Les modèles obtenus à partir de la méthode LDR sont meilleurs que ceux obtenus par planification aléatoire dans 91 % des cas.

### Comparaison entre plan D-optimal et plan LDR

Comme pour le processus de Friedman, le modèle obtenu à partir du plan LHS initial nous permet de planifier de nouvelles bases par la méthode D-optimale et par la méthode LDR. Le test de comparaison des deux méthodes consiste à entraîner 1000 modèles, avec 1000 initialisations différentes sur chacune des bases. Chacun des 1000 modèles est le meilleur modèle en apprentissage parmi 30 modèles initialisés différemment. On calcul l'erreur de généralisation de chacun des 1000 modèles à l'aide d'un ensemble de test comportant 20 000 exemples. La figure 5.17

représente les histogrammes des erreurs de généralisation de ces modèles pour la D-optimalité et la méthode LDR.

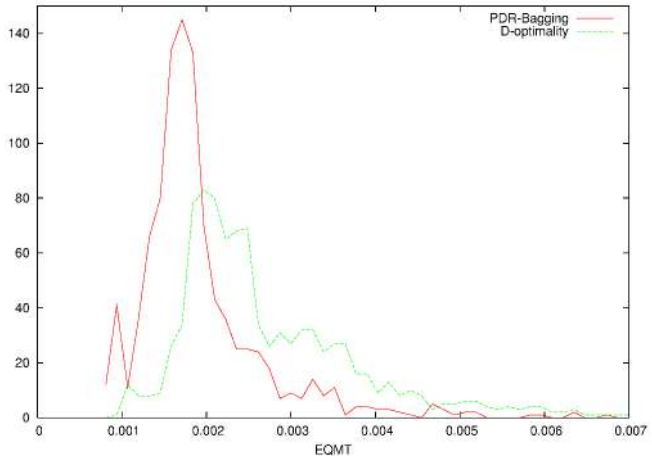



FIG. 5.17 – Histogramme des 1000 erreurs de généralisation pour la planification D-optimale et la planification LDR. Nous avons créé 1000 modèles (chacun d'eux étant le meilleur modèle en apprentissage parmi 30). Nous avons calculé l'erreur de généralisation sur chacun d'eux. Cette figure représente l'histogramme de ces 1000 valeurs pour la D-optimalité et pour la méthode LDR.

Pour le processus considéré, les modèles construits sur la base LDR ont de meilleures performances en généralisation que ceux construits sur le plan D-optimal. Nous pouvons remarquer que la dispersion des erreurs de généralisation de ces modèles est moins importante que pour le processus de Friedman (voir figure 5.11). Ceci peut s'expliquer par le nombre de points de la base finale par rapport à la dimension de l'espace des variables. Les bases finales pour le processus de Friedman contiennent 130 expériences pour un espace des entrées de dimension 5, alors que nous avons des bases finales contenant 160 expériences pour un espace des entrées de dimension 3 pour le processus de Homma et Saltelli.

La figure 5.18 représente la fonction de répartition des erreurs de généralisation. 73 % des modèles entraînés sur la base LDR ont une erreur de généralisation inférieure à $2.10^{-3}$ contre 26% dans le cas des modèles ayant appris sur le plan D-optimal.

### 5.5.4 Zones de replanification

#### Points D-optimaux

La figure 5.19 représente les projections des 60 points D-optimaux replanifiés. Seuls deux points sont situés à l'intérieur du cube expérimental. Tous les autres points replanifiés sont situés sur les sommets, sur les arêtes ou sur les faces de ce cube.

#### Points LDR

La figure 5.20 représente les projections des 60 points LDR replanifiés.

La similitude est très forte. Dans les deux cas, on note une faible présence de points à l'intérieur du domaine d'étude et une forte présence sur les faces et les arêtes du domaine. Notons, néanmoins, que la méthode LDR conduit à replanifier les 8 sommets au lieu de 4 replanifiés par la D-optimalité (voir figure 5.21).

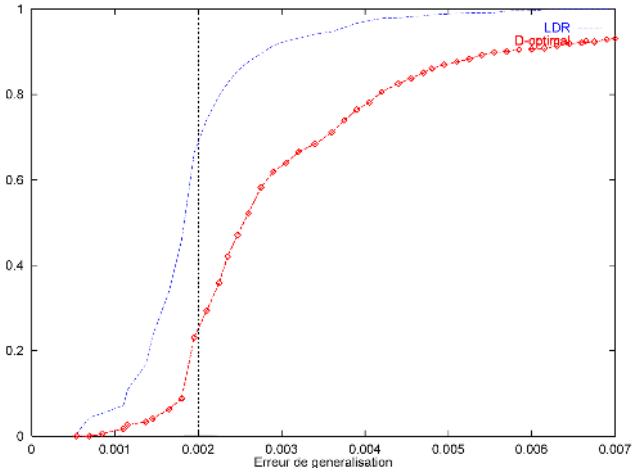



FIG. 5.18 – Fonction de répartition de l'erreur de généralisation des modèles construits à partir des bases D-optimale et LDR. Certains modèles construits à partir du plan D-optimal ont obtenu une erreur de généralisation supérieure à $7.10^{-3}$ que nous n'avons pas reportée sur les graphiques. Ceci explique pourquoi la fonction de répartition pour la D-optimalité n'atteint pas 100 % pour une erreur de généralisation de $7.10^{-3}$.

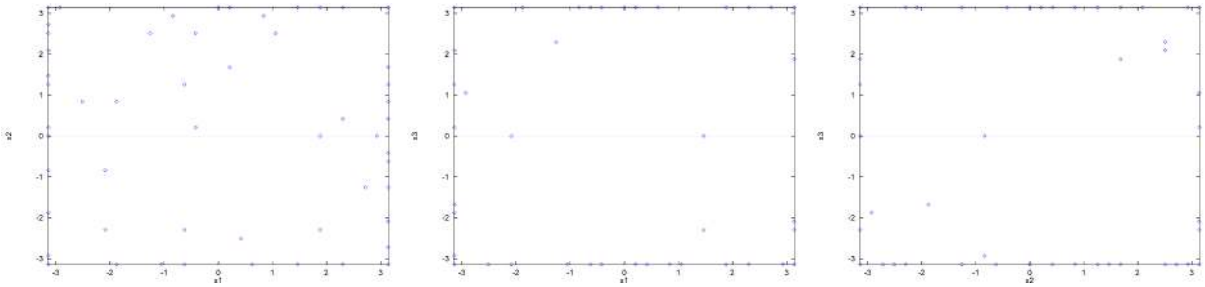


FIG. 5.19 – Plan D-optimal final vu en projection.

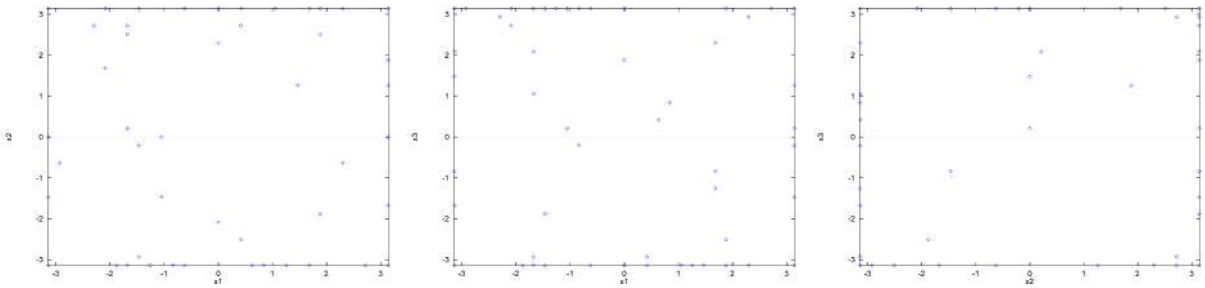


FIG. 5.20 – Plan LDR final vu en projection.

| | **Intérieur** | **Face** | **Arête** | **Sommet** |
|---|---|---|---|---|
| **D-optimal** | 3.33 % | 41.67 % | 48.33 % | 6.67 % |
| **LDR** | 8.33 % | 46.67 % | 31.67 % | 13.33 % |

FIG. 5.21 – Type de point replanifié pour les deux méthodes.

### 5.5.5 Temps de calcul

Ce paragraphe fournit des ordres de grandeurs de temps de calcul pour mener la campagne de planification pour le modèle de Homma et Saltelli. Pour planifier un point, il faut tout d'abord entraîner plusieurs réseaux de neurones, afin de choisir le meilleur en apprentissage. Le vecteur initial des paramètres de ce modèle est ensuite utilisé pour initialiser tous les réseaux de neurones construits à partir des répliques bootstrap de la base d'apprentissage. Il faut réaliser 30 apprentissages pour effectuer cette étape de choix de modèle. La deuxième étape de la méthode consiste à entraîner 200 réseaux de neurones sur des répliques bootstrap de la base d'apprentissage courante. Il faut donc réaliser 200 apprentissages, ainsi que le calcul de la variance des prédictions, pour effectuer cette deuxième étape. Il faut donc au total 230 apprentissages, et le calcul de la variance des prédictions pour planifier un point.

La planification d'un point demande environ 7 minutes sur un ordinateur conventionnel muni d'une unité centrale Pentium IV. Ce temps est négligeable par rapport au temps requis par un code numérique pour le calcul de la réponse correspondante.

### 5.5.6 Évolution de l'erreur de généralisation en fonction du nombre d'exemples

#### Évolution D-optimale et LDR

Le procédé de comparaison que nous avons mis en oeuvre est différent de celui du cas précédent. Nous avions visualisé l'erreur de généralisation pour chaque point replanifié. Étant donné que la variabilité des performances des modèles en fonction de l'initialisation de leurs paramètres est très grande pour le processus de Homma et Saltelli (voir paragraphe 5.14), nous avons réalisé l'histogramme des erreurs de généralisation des 1000 réseaux de neurones construits à partir de la base LHS initiale et l'avons comparé aux histogrammes des erreurs de généralisation des modèles construits à partir de la base D-optimale de 160 points, la base LDR de 160 points ainsi qu'une base LHS de 160 points (voir figure 5.22).

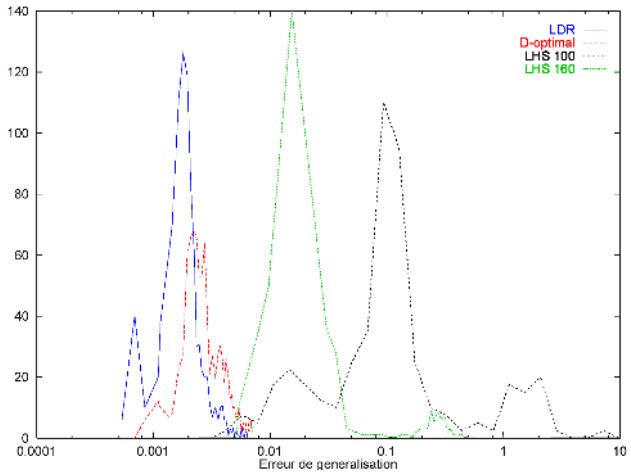



FIG. 5.22 – Histogramme des erreurs de généralisation (en echelle log) de 1000 modèles entrainés sur la base initiale LHS commune aux deux stratégies et les histogrammes des erreurs de généralisation de 1000 modèles entrainés sur une base LHS de 160 points et sur les bases D-optimale et LDR finales.

Pour s'affranchir du problème de la forte influence de l'initialisation des paramètres des

modèles sur leurs performances, nous avons visualisé l'évolution de l'erreur de généralisation en fonction du nombre d'exemples replanifiés pour des modèles linéaires en leurs paramètres.

**Evolution dans le cadre de modèles linéaires**

Comme nous l'avons précisé en début de chapitre, la variance des prédictions peut aussi être estimée par d'autres méthode de ré-échantillonage que le bootstrap, le leave-one-out par exemple. Lorsque les modèles sont linéaires par rapport aux paramètres, la variabilité des paramètres peut être calculée explicitement. C'est pourquoi nous avons utilisé des modèles linéaires en les paramètres (polynômes de degré 6, possédant 84 paramètres) sur le processus de Homma et Saltelli, et avons replanifié le point de variance des prédictions maximale calculée par leave-one-out. Nous allons ensuite comparer la qualité des bases LDR à la qualité des bases LHS de même taille.

Rapellons que les modèles que nous utilisons sont de la forme :

$$f(\mathbf{x}, \boldsymbol{\theta}) = \sum_{k=0}^{p} \theta_k \phi_k(\mathbf{x}) = \phi(\mathbf{x})' \boldsymbol{\theta}$$

La valeur des paramètres $\theta_k$ est obtenue par la méthode des moindres carrés, que nous avons décrite au premier chapitre. Nous avons vu, au chapitre 2 (equation (2.12)) que la variance des paramètres d'un modèle linéaire estimée par leave-one-out s'écrit :

$$s^2_{loo}(\boldsymbol{\theta}) = \mathbf{X}^+ \boldsymbol{\Lambda} (\mathbf{X}^+)'$$

où $\mathbf{X}^+ = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$ est la pseudo inverse de $\mathbf{X}$, et $\boldsymbol{\Lambda}$ est la matrice d'élement $\Lambda_{ij} = \frac{1}{N}\alpha_i^2 \delta_{ij} - \frac{1}{N^2}\alpha_i \alpha_j$ avec $\alpha_i = r_i/(1 - h_{ii})$.

La variance des prédictions au point $\mathbf{x}$ s'exprime comme :

$$s^2_{loo}(f(\mathbf{x}, \boldsymbol{\theta})) = \phi(\mathbf{x})' \mathbf{X}^+ \boldsymbol{\Lambda} \mathbf{X}^{+'} \phi(\mathbf{x})$$

Le point que l'on replanifie est, parmi un ensemble de points candidats, celui pour lequel la variance est maximale :

$$\mathbf{x}_{new} = ArgMax_{\mathbf{x} \in U} ||\phi(\mathbf{x})' \mathbf{X}^+ \boldsymbol{\Lambda}^{1/2}||$$

Nous avons comparé l'erreur de généralisation de modèles construits à partir des bases obtenues par la méthode LDR leave-one-out et celles obtenues par des bases LHS pour des tailles

allant de 100 à 250 expériences. Étant donné que la méthode LHS est aléatoire et afin d'obtenir une comparaison plus robuste entre les deux méthodes, nous avons utilisés 50 bases LHS différentes pour chaque taille et nous avons calculé la moyenne et l'écart-type de l'erreur de généralisation dans chaque cas.

La figure 5.23 montre l'évolution de l'erreur de généralisation des modèles construits à partir des bases LDR ainsi que l'évolution de la moyenne et de l'écart-type de l'erreur de généralisation des modèles construits à partir des bases LHS.

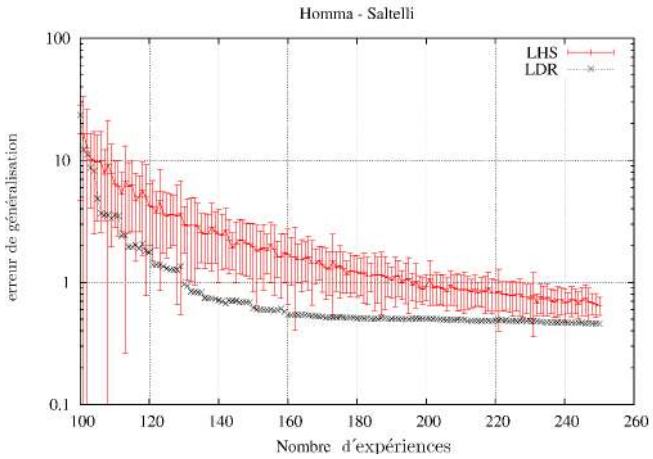



FIG. 5.23 – Comparaison entre les bases LHS et LDR. Nous avons approché la fonction de Homma et Saltelli par des polynômes de degré 6. La variance des prédictions est estimée par leave-one-out virtuel. Pour la stratégie LHS, pour toutes les itérations $k = 1, ..., 150$, 50 bases LHS de taille $100 + k$ ont été créées. Nous avons reporté la moyenne et l'écart-type de l'erreur de généralisation des modèles construits sur ces bases. Pour la méthode LDR, un seul point est ajouté à chaque itération. Le point choisi est celui pour lequel la variance des prédictions est maximale.

Comme on pouvait s'y attendre, la méthode LDR leave-one-out est plus efficace que LHS. Pour des bases de même taille, l'erreur de généralisation des modèles construits à partir des bases LDR est plus faible que la moyenne des erreurs obtenues à partir de LHS d'au moins un écart-type de l'erreur de généralisation LHS. De plus, cette étude nous a permis de visualiser l'évolution de l'erreur de généralisation en fonction du nombre de points ajoutés en dehors de toute considération de choix de modèles étant donné que nous utilisons des polynômes. C'est d'ailleurs pour cela que l'évolution est beaucoup plus régulière que dans le cas des réseaux de neurones (voir figure 5.13 pour le processus de Friedman).

## 5.6 Conclusion

La méthode d'apprentissage actif LDR, que nous proposons pour planifier des expériences numériques pour modèles non linéaires, a été décrite en détail dans ce chapitre. Nous avons présenté des comparaisons entre les capacités de généralisation de modèles qui ont été conçus par apprentissage à partir de bases de données planifiées par LDR et par D-optimalité locale. Deux processus générateurs des données ont été utilisés : le processus de Friedman et celui de Homma et Saltelli. Les résultats concernant le premier processus ne mettent pas en évidence de différences significatives entre les qualités des bases créées par les deux méthodes. En revanche, sur le processus de Homma et Saltelli, les capacités de généralisation de modèles conçus à partir de la base d'exemples LDR étaient supérieures à celles de modèles construits à partir de la base D-optimale.

# Conclusion Générale

Nous nous sommes attachés, dans cette thèse, à développer une procédure de planification d'expériences dans le contexte de la simulation numérique. Nous avons présenté une méthode de planification d'expériences simulées fondée sur l'apprentissage actif. Cette discipline faisant partie de la théorie de l'apprentissage statistique, nous avons consacré le premier chapitre de ce mémoire à décrire les fondements de cette théorie.

Une première partie de mon travail de thèse a été de déterminer l'importance d'un exemple dans le processus d'apprentissage par des méthodes statistiques de ré-échantillonnage. Déterminer une mesure de l'influence des exemples nous paraissait être un premier pas vers la planification d'expériences. Dans le cadre de modèles linéaires par rapport aux paramètres et pour des données bruitées, la variance des prédictions des modèles est corrélée aux leviers. Le coefficient de corrélation est la variance du bruit expérimental. Cependant, pour des données déterministes, cette corrélation n'est plus valable puisque le coefficient de corrélation (le bruit expérimental) n'existe pas. C'est pourquoi nous avons estimé la variance des prédictions par la technique de ré-échantillonnage du bootstrap. Même si les données sont déterministes, le ré-échantillonnage crée des bases d'apprentissage différentes, chacune d'elles permet de créer un modèle différent et, dans ce cas, la variance des prédictions n'est pas nulle. Nous avons ensuite étudié les relations existant entre la variance des prédictions estimée par bootstrap et les leviers. Les résultats obtenus dans le cadre de cette étude ont montré :

- que l'on peut déterminer l'importance d'un exemple dans le processus d'apprentissage par le calcul de la variance des prédictions de modèles créés à partir de répliques bootstrap de la base d'apprentissage.

- que la mesure de l'importance d'un exemple à partir de la variance des prédictions prend en considération les informations relatives à la sortie du processus à modéliser. En effet, la réponse d'un modèle est dépendante du placement des expériences de la base d'apprentissage dans l'espace des variables et des leurs sorties associées. Contrairement aux leviers calculés uniquement à partir de la répartition des points dans l'espace des variables (méthode *a priori*), la variance des prédictions est conditionnée par le ré-échantillonnage qui s'effectue sur les données d'entrée et de sortie (méthode *a posteriori*).

Nous avons consacré le chapitre 3 à la problématique de choix de modèles. Cette problématique est directement liée à l'estimation de l'erreur de généralisation. Nous avons présenté une méthode d'estimation de l'erreur de généralisation pour des modèles linéaires en leurs paramètres fondée sur l'analyse spectrale de la matrice d'information, qui nous a permis d'établir un lien intéressant entre l'apprentissage statistique et la méthodologie des plans d'expériences (avec la A-optimalité). Pour les modèles non linéaires en les paramètres de type réseaux de neurones, nous avons utilisé des méthodes de ré-échantillonnage pour estimer l'erreur de généralisation. Nous avons validé, sur des exemples simples, l'estimation de l'erreur de généralisation par ces méthodes en les comparant à une très bonne estimation de l'erreur de généralisation établie sur une base de test comportant plusieurs milliers d'exemples.

Après avoir décrit les principaux plans d'expériences au chapitre 4, nous avons présenté la méthode LDR (Learner Disagreement by experiment Resampling) que nous proposons dans cette thèse pour planifier des expériences numériques. Nous avons utilisé deux processus générateurs des données, le processus de Friedman et celui de Homma et Saltelli pour présenter des comparaisons entre les capacités de généralisation des modèles conçus à partir de bases d'exemples

planifiées par LDR et par D-optimalité locale. Nous avons remarqué que les capacités de généralisation des modèles construits à partir de la base d'exemples LDR étaient supérieures à celles des modèles construits sur la base D-optimale pour le processus de Homma et Saltelli. Initialement, les plans d'expériences sont conçus pour des modèles linéaires par rapport aux paramètres et pour des données bruitées. Nous les avons utilisés dans le cadre d'expériences simulées, pour lequel les hypothèses concernant le bruit de mesure ne sont pas vérifiées, et pour des modèles non linéaires en leurs paramètres. Néanmoins, cette étude a permis de mettre en évidence l'efficacité des plans D-optimaux dans ce cadre expérimental particulier.

La méthode LDR que nous proposons offre de bons résultats. Il faudrait cependant établir l'influence du plan LHS initial sur la qualité des bases finales obtenues à partir des deux méthodes, en reconduisant l'ensemble de la procédure sur différentes bases LHS initiales. Une perspective serait d'utiliser les méthodes de discrépance pour créer la base d'apprentissage initiale nécessaire pour l'initialisation des deux méthodes. La qualité des bases planifiées par la D-optimalité et par LDR doit dépendre de la qualité de la base d'apprentissage initiale.

Nous avons également rencontré certains problèmes lorsque l'on crée, par ré-échantillonnage, un échantillon peu représentatif des données. Le modèle construit à partir de cet échantillon risque de mal représenter le processus à modéliser. Ce modèle *outlier* risque de biaiser la statistique de la population de modèles. Une solution est d'effectuer le calcul de la variance des prédictions sur les meilleurs modèles de la population. Une autre solution serait d'envisager d'autres mesures de divergence plus robustes que la variance comme par exemple l'entropie.

# Annexe A

# Calcul du gradient de la fonction de coût

## A.1 Calcul du gradient

Le vecteur gradient de la fonction de coût peut être calculé, dans le cadre des réseaux de neurones, par un calcul direct. Néanmoins, une implémentation plus judicieuse du calcul du gradient peut être obtenue à l'aide de l'*algorithme de rétropropagation* proposé par [Rumelhart & McClelland 86], [Werbos 74]. La présentation de ces algorithmes est extraite de [Dreyfus *et al.* 04] et [Gaudier 99].

Pour ces deux algorithmes, nous considérons un réseau de neurones de type MLP. Rappelons que le neurone $i$ calcule une grandeur $y_i$, non linéaire par rapport à son potentiel $p_i$ qui est une somme des entrées $x_j$ pondérée par les paramètres $\theta_{ij}$.

$$y_i = f(\sum_{j=1}^{n_i} \theta_{ij} x_j) = f(p_i)$$

Les $n_i$ entrées du neurone $i$ peuvent être les entrées du réseau ou les sorties d'autres neurones. $x_j$ désignera soit la sortie $y_j$ du neurone $j$ soit l'entrée $j$ du réseau.

Le fonction de coût dont on cherche à évaluer le gradient est :

$$J(\boldsymbol{\theta}) = \sum_{k=1}^{N} (y^k - f(\mathbf{x}^k, \boldsymbol{\theta}))^2 = \sum_{k=1}^{N} J^k(\boldsymbol{\theta}) \tag{A.1}$$

où $y^k$ est la sortie du processus à modéliser en réponse à l'exemple $k$.

La fonction de coût est donc décomposée en somme de coûts partiels $J^k(\boldsymbol{\theta})$. Il suffit donc d'évaluer le gradient du coût partiel relatif à l'exemple $k$ et de faire la somme sur l'ensemble des exemples pour obtenir le gradient de la fonction de coût totale.

### Calcul du gradient dans le sens direct

Il faut distinguer deux cas dans le cadre du calcul du gradient dans le sens direct.

– pour un neurone $m$ qui reçoit $x_j^k$ directement de l'entrée $j$ du réseau ou de la sortie du neurone $j$ :

$$\Big(\frac{\partial y_m}{\partial \theta_{mj}}\Big)_k = \Big(\frac{\partial y_m}{\partial p_m}\Big)_k \Big(\frac{\partial p_m}{\partial \theta_{mj}}\Big)_k = f'(p_m^k) x_j^k$$

où
  – $\Big(\frac{\partial y_m}{\partial \theta_{mj}}\Big)_k$ désigne la valeur de la dérivée partielle de la sortie du neurone $m$ par rapport au paramètre $\theta_{mj}$ lorsque les entrées du réseau sont celles qui correspondent à l'exemple $k$.

  – $\Big(\frac{\partial y_m}{\partial p_m}\Big)_k$ désigne la valeur de la dérivée partielle de la sortie du neurone $m$ par rapport au potentiel du neurone $m$ lorsque les entrées du réseau sont celles qui correspondent à l'exemple $k$.

- $\left(\frac{\partial p_m}{\partial \theta_{mj}}\right)_k$ correspond à la valeur de la dérivée partielle du potentiel du neurone $m$ par rapport au paramètre $\theta_{mj}$ lorsque les entrées du réseau sont celles qui correspondent à l'exemple $k$.

- $x_j^k$ est la valeur de l'entrée $j$ du réseau pour l'exemple $k$.

– pour un neurone $m$ qui reçoit l'information $x_j^k$ de l'entrée $j$ du réseau ou de la sortie du neurone $j$ par l'intermédiaire d'autres neurones du réseau, situés entre les entrées et le neurone $m$ :

$$\left(\frac{\partial y_m}{\partial \theta_{ij}}\right)_k = \left(\frac{\partial y_m}{\partial p_m}\right)_k \left(\frac{\partial p_m}{\partial \theta_{ij}}\right)_k = f'(p_m^k) \sum_l \left(\frac{\partial p_m}{\partial y_l}\right)_k \left(\frac{\partial y_l}{\partial \theta_{ij}}\right)_k = f'(p_m^k) \sum_l \theta_{ml} \left(\frac{\partial y_l}{\partial \theta_{ij}}\right)_k$$

où $l$ désigne les neurones intermédiaires situés entre les entrées et le neurone $m$.

Ces deux relations permettent de calculer les dérivées de la sortie de chaque neurone par rapport aux paramètres qui ont une influence sur cette sortie, à partir des entrées du réseau jusqu'à la sortie. Une fois toutes ces dérivées calculées, on peut établir le gradient de la fonction de coût partielle :

$$\left(\frac{\partial J^k}{\partial \theta_{ij}}\right)_k = \left(\frac{\partial}{\partial \theta_{ij}} \left[(y_p^k - f(\mathbf{x}, \boldsymbol{\theta}))^2\right]\right)_k = -2(y_p^k - f(\mathbf{x}^k, \boldsymbol{\theta})) \left(\frac{\partial f(\mathbf{x}, \boldsymbol{\theta})}{\partial \theta_{ij}}\right)_k$$

On obtient le gradient de la fonction de coût en faisant la somme des fonctions de coût partielles sur l'ensemble des exemples.

**Algorithme de rétropropagation**

Cette implémentation permet de faire le calcul du gradient avec moins d'opérations que pour celui obtenu dans le sens direct. Il faut préciser que ce n'est pas un algorithme d'apprentissage mais uniquement un algorithme de calcul de gradient qui sert à l'apprentissage.

L'algorithme de rétropropagation du gradient repose sur l'utilisation répétée de la règle des dérivées composées. Etant donné que le coût partiel ne dépend du paramètre $\theta_{ij}$ que par l'intermédiaire de la valeur de la sortie du neurone $i$ qui est une fonction uniquement du potentiel du neurone $i$, on peut écrire :

$$\left(\frac{\partial J^k}{\partial \theta_{ij}}\right)_k = \left(\frac{\partial J^k}{\partial p_i}\right)_k \left(\frac{\partial p_i}{\partial \theta_{ij}}\right)_k = \delta_i^k x_j^k \tag{A.2}$$

avec

- $\left(\frac{\partial J^k}{\partial p_i}\right)_k$ correspond à la valeur du gradient du coût partiel par rapport au potentiel du neurone $i$ avec l'exemple $k$ en entrée du réseau.

– $\left(\frac{\partial p_i}{\partial \theta_{ij}}\right)_k$ correspond à la valeur de la dérivée partielle du potentiel du neurone $i$ par rapport au paramètre $\theta_{ij}$ avec l'exemple $k$ en entrée du réseau.

– $x_j^k$ est la valeur de l'entrée $j$ du neurone $i$ lorsque les entrées du réseau sont celles qui correspondent à l'exemple $k$.

Il faut donc évaluer les quantités $\delta_i^k$. Le calcul peut être mené judicieusement d'une manière récursive en menant les calculs depuis la sortie du réseau vers ses entrées.

Nous devons distinguer deux cas :
– pour le neurone de sortie $i$ :

$$\delta_i^k = \Big(\frac{\partial J^k}{\partial p_i}\Big)_k = \Big(\frac{\partial}{\partial p_i}[(y^k - f(\mathbf{x}, \boldsymbol{\theta})^2]\Big)_k = -2f(\mathbf{x}^k, \boldsymbol{\theta})\Big(\frac{\partial f(\mathbf{x}, \boldsymbol{\theta})}{\partial p_i}\Big)_k$$

Cette relation peut être simplifiée dans le cadre du neurone de sortie en régression, car dans ce cas, le neurone de sortie est linéaire et l'expression se réduit à :

$$\delta_i^k = -2f(\mathbf{x}^k, \boldsymbol{\theta})$$

– pour un neurone caché $i$ :

La fonction de coût ne dépend du potentiel du neurone $i$ que par l'intermédiaire des potentiels des neurones $m$ qui reçoivent la valeur de la sortie du neurone $i$. Ces neurones $m$ sont ceux situés entre le neurone $i$ et la sortie du réseau.

$$\Big(\frac{\partial J^k}{\partial p_i}\Big)_k = \sum_m \Big(\frac{\partial J^k}{\partial p_m}\Big)_k \Big(\frac{\partial p_m}{\partial p_i}\Big)_k = \sum_m \delta_m^k \Big(\frac{\partial p_m}{\partial p_i}\Big)_k \tag{A.3}$$

or $p_m^k = \sum_i \theta_{mi} x_i^k = \sum_i \theta_{mi} f(p_i^k)$, nous avons alors : $\left(\frac{\partial p_m}{\partial p_i}\right)_k = \theta_{mi} f'(p_i^k)$

On obtient finalement la relation :

$$\delta_i^k = \sum_m \delta_m^k \theta_{mi} f'(p_i^k) = f'(p_i^k) \sum_m \delta_m^k \theta_{mi}$$

Les quantités $\delta_i^k$ peuvent être obtenues récursivement en parcourant le graphe des connexions de la sortie vers les entrées du réseau.

La rétropropagation nécessite l'évaluation d'un gradient par neurone alors que le calcul dans le sens direct requiert l'évaluation d'un gradient par connexion. Le nombre d'évaluation de gradient dans le sens direct est donc plus important que dans le cas de la rétropropagation.

# Annexe B

# Algorithme d'optimisation de Levenberg-Marquardt

## B.1 L'algorithme d'apprentissage de Levenberg-Marquardt

Les algorithmes d'apprentissage des réseaux de neurones par utilisation du gradient, sont fondées sur les méthodes classiques de descente, comme par exemple le gradient stochastique adapté aux grandes bases d'exemples ou le gradient conjugué et les méthodes quasi-newton aux convergences plus rapides. Pour plus de détails, on pourra consulter notamment [Dreyfus *et al.* 04], [Cichocki & Unbehauen 93]. A noter le problème posé par les minima locaux. En effet pour des modèles non linéaires par rapport aux paramètres comme les réseaux de neurones, la fonction de coût n'étant pas convexe, la solution optimale ne vérifie que les conditions locales du minimum.

Dans cette section, nous allons uniquement présenter la méthode quasi-newton de Levenberg-Marquardt [Levenberg 44] [Marquardt 63] particulièrement adaptée à la minimisation de fonction coût de type moindres carrés, que nous avons utilisée comme algorithme d'apprentissage des réseaux de neurones.

Rappelons le coût global $J$ définis par :

$$J = \frac{1}{N}\sum_k (f(\mathbf{x}^k, \boldsymbol{\theta}) - y^{*k})^2$$

En notant $J_k = f(\mathbf{x}^k, \boldsymbol{\theta}) - y^{*k}$ le résidu de l'exemple $k$ et $\mathbf{J}' = (J_1, J_2, \ldots)$ le vecteur des coûts partiels :

$$\mathbf{J} = \frac{1}{N}\sum_k J_k^2 = \frac{1}{N}\mathbf{J}'\mathbf{J}$$

Soient $\nabla\mathbf{J}$ et $\nabla^2\mathbf{J}$ le gradient et la matrice hessienne de $C$, un développement au second ordre autour de la valeur $\boldsymbol{\theta}_0$ s'écrit :

$$J(\boldsymbol{\theta}_0 + \delta\boldsymbol{\theta}) \simeq J(\boldsymbol{\theta}_0) + \delta\boldsymbol{\theta}'\nabla J(\boldsymbol{\theta}_0) + \frac{1}{2}\delta\boldsymbol{\theta}'\nabla^2 J(\boldsymbol{\theta}_0)\delta\boldsymbol{\theta}$$

La variation $\delta\boldsymbol{\theta}$ minimisant la variation (méthode de Newton) du coût vérifie :

$$\nabla^2 J(\boldsymbol{\theta}_0)\delta\boldsymbol{\theta} = -\nabla J(\boldsymbol{\theta}_0)$$

Lorsque la fonction de coût s'exprime sous forme d'une moyenne quadratique, il vient :

$$\begin{aligned}
\nabla J(\boldsymbol{\theta}_0)_i &= \frac{2}{\mathtt{N}}\sum_k J_k \frac{\partial f(\mathbf{x}_k, \boldsymbol{\theta})}{\partial\theta_i} \\
\nabla^2 J(\boldsymbol{\theta}_0)_{ij} &= \frac{2}{\mathtt{N}}\sum_k \frac{\partial f(\mathbf{x}_k, \boldsymbol{\theta})}{\partial\theta_i}\frac{\partial f(\mathbf{x}_k, \boldsymbol{\theta})}{\partial\theta_j} + J_k\frac{\partial^2 f(\mathbf{x}_k, \boldsymbol{\theta})}{\partial\theta_i\partial\theta_j}
\end{aligned}$$

Dans l'algorithme de Levenberg - Marquardt (Gauss-Newton) le second terme de la matrice hessienne est négligé[18].

$$\nabla^2 J(\theta_0)_{ij} \simeq \frac{2}{\mathtt{N}} \sum_k \frac{\partial f(\mathbf{x}_k, \theta)}{\partial \theta_i} \frac{\partial f(\mathbf{x}_k, \theta)}{\partial \theta_j}$$

En notant par $\mathbf{Z}$ la matrice des éléments de sensibilité du premier ordre, la méthode aboutit à l'approximation de la résolution du système suivant :

$$\begin{aligned} Z_{ij} &= \frac{\partial f(\mathbf{x}_i, \boldsymbol{\theta})}{\partial \theta_j} \\ (\mathbf{Z}'\mathbf{Z})\delta\boldsymbol{\theta} &= -\mathbf{Z}'\mathbf{J} \end{aligned}$$

A noter que la résolution de ce système correspond à la solution des moindres carrés $||\mathbf{Z}\delta\boldsymbol{\theta} + J||^2$ et est celui des **équations normales**. Dans les cas où la matrice $\mathbf{Z}$ est mal conditionnée, ou de rang non maximal, il est nécessaire de régulariser ce système. Cela est généralement obtenu via un paramètre $\lambda \geq 0$ de régularisation :

$$(\mathbf{Z}'\mathbf{Z} + \lambda\mathbf{I})\delta\boldsymbol{\theta} = -\mathbf{Z}'\mathbf{J} \tag{B.1}$$

La matrice $\mathbf{Z}'\mathbf{Z}+\lambda\mathbf{I}$ étant symétrique définie positive, la nouvelle direction reste une direction de descente ($\delta\boldsymbol{\theta}'(\mathbf{Z}'\mathbf{Z} + \lambda\mathbf{I})\delta\boldsymbol{\theta} > 0$). On peut donc déterminer le pas $\eta$ du gradient de façon à réduire le cout $J$ c'est à dire de façon à obtenir $J(\boldsymbol{\theta} - \eta\delta\boldsymbol{\theta}) < J(\boldsymbol{\theta})$.

Cet algorithme est relativement performant. Il ne nécessite que le calcul des dérivées premières obtenues pour les réseaux de neurones par rétro-propagation. Pour des valeurs importantes du terme de régularisation $\lambda$ il se comporte comme la méthode du gradient simple. Pour des valeurs plus petites, l'algorithme se rapproche de la méthode de Newton et présente donc d'excellents performances en convergence en réduisant le nombre d'itétrations nécessaire.

Le système B.1 peut être résolu par les approches itératives de type gradients conjugués (problème de grande taille) ou bien par les méthodes directes (pour les problèmes de petite taille) de type `Choleski`. A noter égalemement les méthodes de décomposition de type **QR** ou **SVD** en valeurs singulières pour les problèmes de petite taille et particulièrement adaptées à la résolution des moindres carrés.

[18] On retrouve la même approximation à partir d'un développement au premier ordre de la fonction $f(\mathbf{x}, \boldsymbol{\theta})$ par rapport à $\boldsymbol{\theta}$.

# Annexe C

# Article soumis à IEEE Transactions on Neural Networks

# Towards The Optimal Design of Numerical Experiments.

Stéphane Gazut[1], Jean-Marc Martinez[1], Gérard Dreyfus[2] & Yacine Oussar[2]

[1] Commissariat à l'Energie Atomique - Centre d'étude de Saclay
Direction à l'Energie Nucléaire - Département de Modélisation des Systèmes et Structures
Service Fluides numériques, Modélisation et Etudes - Laboratoire d'Etudes Thermiques des Réacteurs
DEN / DM2S / SFME / LETR
91191 Gif-sur-Yvette Cedex
E-mail : *stephane.gazut@cea.fr* and *jean-marc.martinez@cea.fr*

[2] Ecole Supérieure de Physique et de Chimie Industrielles de la Ville de Paris
(ESPCI-Paristech)
Laboratoire d'Electronique (UMR CNRS 7084)
10, rue Vauquelin
75231 Paris Cedex 05
E-mail : *gerard.dreyfus@espci.fr* and *yacine.oussar@espci.fr*

**Abstract**

The present paper addresses the problem of the optimal design of numerical experiments for the construction of nonlinear surrogate models. We describe a new method, called LDR for Learner Disagreement from experiment Resampling, which borrows ideas from active learning and from resampling methods : the analysis of the divergence of the predictions provided by a population of models, constructed by resampling, allows an iterative determination of the point of input space where a numerical experiment should be performed in order to improve the accuracy of the predictor. The LDR method is illustrated on neural network models with bootstrap resampling, and on orthogonal polynomials with leave-one-out resampling. Other methods of experimental design such as random selection and D-optimal selection are investigated on the same benchmark problems.

**keywords**

Active Learning, Bagging, Bootstrap, Neural Networks, D-optimality.

## C.1 Introduction

Although numerical simulation tends to be ubiquitous in present-day engineering, computation time often limits its use, despite the ever-increasing power of computers. A common technique for circumventing that limitation is the design of *surrogate models*, i.e. analytical functions that

approximate the input-output mapping performed by the simulation model. Still, the estimation of the parameters of the surrogate models requires the availability of results obtained by the simulation model that it is intended to approximate ; therefore, whenever numerical experiments are costly, it is important to select them as efficiently as possible in order to minimize their number. In statistics, the selection of experiments is known as *optimal experimental design* (OED), see for instance [1] and [2], while it is known as *active learning* in the machine learning literature.

Optimal experimental design has been widely developed for models that are linear in their parameters, such as polynomials. The observations of a given quantity are assumed to be realizations of a random variable that is the sum of a deterministic function (the regression function, assumed to be linear in its parameters) and of a random variable with zero mean. By contrast, in the present work, the existence of the latter random variable is not assumed : in other words, repeated experiments will provide identical results. Such is the case when the data to be modeled is generated by a deterministic computer simulation. Moreover, we relax the assumption that the model is linear in its parameters. Therefore, we describe a generic alternative approach to experimental design, based on resampling techniques.

Section C.2 and section C.3 are intended to put optimal experimental design and active learning into the perspective of the present work. The two subsequent sections describe two variants of the method that we advocate in the context of numerical experiments. Finally, those approaches are compared on classical benchmark problems.

## C.2 Background : D-optimality in experimental design

The mainstream development of optimal experimental design dealt with linear-in-their-parameters models, starting with the work of Kiefer [3], Kiefer and Wolfowitz [Kiefer 59], Fedorov [1] or Wynn [5]. Vila [Vila 91], MacKay [MacKay 92a] and Cohn [8], and more recently Issanchou and Gauchi [Issanchou & Gauchi 04] and Witczak [Witczak 06], described applications of optimal experimental design techniques to the training of neural networks.

### C.2.1 Training models from data

We consider an unknown function $y(\mathbf{x})$ in a domain $U \subset R^d$. We denote by $\mathcal{L} = \{(y_k, \mathbf{x}_k), k = 1, \ldots, n\}$ a finite set of observations, where $\mathbf{x}$ is drawn from a probability distribution $p(\mathbf{x})$, and where $y_k = y(\mathbf{x}_k)$.

We consider a family of parameterized functions $f(\mathbf{x}, \boldsymbol{\theta})$, within which we seek the "best" approximation of the unknown function $y(\mathbf{x})$, given the available data $\mathcal{L}$. To that effect, the loss function $l(y(\mathbf{x}), f(\mathbf{x}, \boldsymbol{\theta})) = \|y(\mathbf{x}) - f(\mathbf{x}, \boldsymbol{\theta})\|^2$ is defined, which expresses the discrepancy between function $y(\mathbf{x})$ and its approximation $f(\mathbf{x}, \boldsymbol{\theta})$. The parameters of the model are estimated by minimizing a cost function which is the sum, over all examples of a data set called training set, of $l(y_k, f(\mathbf{x}_k, \boldsymbol{\theta}))$ ; we denote by $\boldsymbol{\theta}_{\mathcal{L}}$ the vector of parameters for which the cost function is minimum :

$$f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}}) = \texttt{ArgMin}_{f(\mathbf{x},\boldsymbol{\theta})} \sum_{(y_k,\mathbf{x}_k)\in\mathcal{L}} l(y_k, f(\mathbf{x}_k, \boldsymbol{\theta})) \tag{C.1}$$

### C.2.2 The linear framework

In the linear framework, D-optimal experimental design consists in organizing the experiments in order to minimize the variance of the estimated parameters, by maximizing the Fisher Matrix determinant ($det(\mathbf{X'X})$), where $\mathbf{X}$ is the experimental matrix, whose element $x_{ij}$ is the value of variable $j$ observed in experiment $i$. $\mathbf{X}$ is a $(N, p)$ matrix, where $N$ is the number of observations and $p$ is the number of variables. We denote by $\mathbf{X}^+ = (\mathbf{X'X})^{-1}\mathbf{X'}$ the pseudo-inverse of $\mathbf{X}$.

Let $\boldsymbol{\theta}_{\mathcal{L}} = \mathbf{X}^+\mathbf{y}$ be the least-squares estimator of the unknown function parameters for the dataset $\mathcal{L}$. The model $f(\mathbf{x}, \boldsymbol{\theta})$ is postulated to be linear in its parameters. In the probabilistic framework, under the hypothesis of uncorrelated centered residuals with variance $\sigma^2$, the variance-covariance matrix of the parameters is :

$$V(\boldsymbol{\theta}_{\mathcal{L}}) = V(\mathbf{X}^+y) = \mathbf{X}^+V(y)\mathbf{X}^{+'} = \sigma^2(\mathbf{X'X})^{-1} \tag{C.2}$$

The Fisher Matrix $\mathbf{X'X}$ depends on the distribution of the experimental values of the variables. It is therefore natural to seek a distribution of points that reduces the variance of the parameters to the largest extent. Under the additional hypothesis of gaussian residual error, the confidence area of the estimated parameters is a hyperellipsoid centered in $\theta_{\mathcal{L}}$ and defined, for a confidence level $\alpha$, by [Rao & Toutenburg 95] :

$$(\boldsymbol{\theta} - \boldsymbol{\theta}_{\mathcal{L}})\mathbf{X'X}(\boldsymbol{\theta} - \boldsymbol{\theta}_{\mathcal{L}}) \leq \sigma^2\chi^2_\alpha(p) \tag{C.3}$$

where $\chi^2_\alpha(p)$ the Chi-square $\alpha$ quantile with $p$ degrees of freedom.

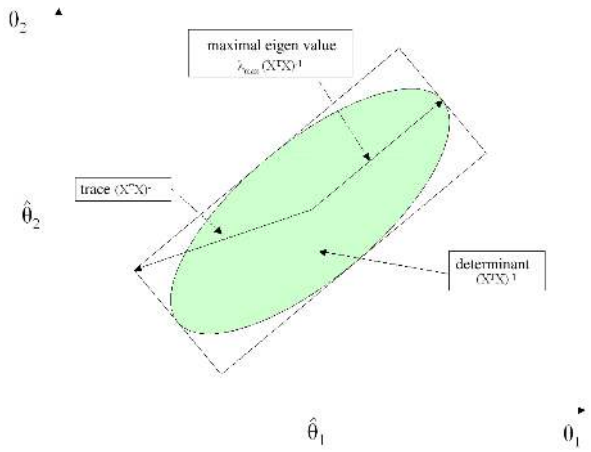



Fig. C.1 – Confidence area for 2 parameters

Many optimality criteria may be considered. We will describe the main optimal experimental design techniques that make use of the spectral properties of the dispersion matrix $(\mathbf{X'X})^{-1}$.

The confidence area, or, more generally, the volume of the confidence ellipsoid as shown on Figure C.1, can be acted upon by decreasing :

– the length of the main axis of the ellipsoid, i.e. the largest eigenvalue of $(\mathbf{X'X})^{-1}$ (E-optimality criterion).

– the sum of the lengths of the axes of the ellipsoid, i.e. the trace of $(\mathbf{X}'\mathbf{X})^{-1}$ (A-optimality criterion).
– the volume of the ellipsoid, i.e. the determinant of $(\mathbf{X}'\mathbf{X})^{-1}$ (D-optimality criterion).

Various algorithms [Mitchell 74], [1] are available for finding exact solutions satisfying the above optimality criteria, for postulated models that are linear in their parameters. In that context, the solution depends only on matrix $\mathbf{X}$ : therefore, it does not depend on the model, insofar as it is postulated to be linear in its parameters. In other words, experimental planning can be performed *prior to modelling* in that context. That is no longer true for models that are nonlinear in their parameters, as will be shown in Section C.2.3, which describes a D-optimal experimental design methodology for such models.

### C.2.3 The non-linear framework

In the non-linear case, useful results are frequently obtained by performing a first-order Taylor expansion of the model, in parameter space, in the neighbourhood of the parameter vector $\boldsymbol{\theta}_{\mathcal{L}}$ for which the least-squares cost function is minimum

$$f(\mathbf{x}, \boldsymbol{\theta}) \simeq f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}}) + \mathbf{Z}(\boldsymbol{\theta}_{\mathcal{L}})(\boldsymbol{\theta} - \boldsymbol{\theta}_{\mathcal{L}}) \tag{C.4}$$

where $\mathbf{Z}$ is the Jacobian matrix of the model

$$[\mathbf{Z}(\mathbf{x}_i, \boldsymbol{\theta})]_{ij} = \Big(\frac{\partial f(\mathbf{x}, \boldsymbol{\theta})}{\partial \theta_j}\Big)_{\mathbf{x}=\mathbf{x_i}} \tag{C.5}$$

$\mathbf{x_i}$ is the vector of variables for the $i$-th observation, and $\theta_j$ is the j-th parameter of model $f(.)$ with parameter vector $\boldsymbol{\theta}$. $\mathbf{Z}$ is an $(N, p)$ matrix, where $N$ is the number of observations and $p$ is the number of parameters of the model. This provides a locally linear approximation of the model, whose variables are the partial derivatives of the model with respect to its parameters. Therefore, the jacobian matrix $\mathbf{Z}$ of the model plays the same role as the experimental matrix $\mathbf{X}$ does for linear-in-their-parameters models. Actually, if the model is linear in its parameters, matrices $\mathbf{X}$ and $\mathbf{Z}$ are identical.

By contrast to matrix $\mathbf{X}$, matrix $\mathbf{Z}$ depends on the parameters of the model. That technique allowed, for instance, the estimation of confidence intervals [13], of the tangent-plane leverages and of the generalization error [Monari & Dreyfus 00] of nonlinear models. In the same spirit, Issanchou and Gauchi [Issanchou & Gauchi 04] proposed, in the homoscedastic case, an optimal experimental planning technique based on the minimization of the approximate volume of the confidence ellipsoid, proportional to $(\mathbf{Z}'\mathbf{Z})$.

Since the Jacobian matrix depends on the parameters of the model, experimental planning cannot be performed prior to modelling. Therefore, a two-step procedure is necessary. Before the construction of the D-optimal design, an initial set of experiments must be available e.g. by Latin Hypercube Sampling (LHS)[19] ; from that initial data set, a first estimate of the parameters of the nonlinear model is obtained, allowing the computation of the Jacobian matrix.

[19] The LHS method was developed to generate a distribution of experiments from a multidimensional distribution [Iman *et al.* 81]. A square grid is a latin square if there is only one sample in each row and each column. A latin hypercube is the generalisation of a latin square in an arbitrary number of dimensions. The LHS sampling provides an efficient sample placement in the input space of variables.

The algorithms that are available for the construction of D-optimal experimental design can be applied simply, replacing matrix $\mathbf{X}$ by matrix $\mathbf{Z}$, in order to obtain D-optimal experiments that can be used in addition to the initial ones. Local D-optimality is often denoted as D($\boldsymbol{\theta}_{\mathcal{L}}$)-optimality, where $\boldsymbol{\theta}_{\mathcal{L}}$ represents the parameter vector for which the least-squares cost function is minimum.

### C.2.4 Algorithmic Construction of D-optimal experimental designs

There are many algorithms for the construction of optimal experimental designs, see for instance [5], [Mitchell 74], or [Atkinson & Donev 89]. We describe here Fedorov's algorithm [1], which is probably the most popular, and easiest to code. The purpose is to select $N$ experiments, in a set of $N_c$ candidates, which maximize the determinant of the Fisher Matrix ($\mathbf{X}'\mathbf{X}$) for linear-in-their-parameters models, or ($\mathbf{Z}'\mathbf{Z}$) for nonlinear-in-their-parameters models.

- First step : choose $N$ experiments randomly in a set of $N_c$ candidate experiments, which are typically the nodes of a "fine" grid.
- Second step : perform all possible exchanges of an experiment $i$ of the initial design with an experiment $j$ of the candidate experiments ; there are $N(N_c - N)$ different exchanges (repeated experiments are not allowed in the context of numerical experiments since repeated numerical experiments yield identical results) ; compute the $N(N_c - N)$ determinants of the corresponding Fisher matrices.
- Third step : perform the exchange that increases the determinant of the Fisher matrix by the largest amount. Iterate to the second step if the termination criterion is not satisfied.

To compute the determinant at the current iteration, the following theorem can be used [Rao & Toutenburg 95] :

After the exchange of $i$ with $j$ at iteration $t$, the new information matrix is :

$$(\mathbf{X}'\mathbf{X})_{[t+1]} = (\mathbf{X}'\mathbf{X})_{[t]} - \mathbf{x}_i\mathbf{x}_i' + \mathbf{x}_j\mathbf{x}_j' \tag{C.6}$$

As a consequence, the determinant is :

$$det((\mathbf{X}'\mathbf{X})_{[t+1]}) = det((\mathbf{X}'\mathbf{X})_{[t]}) \times [1 + \Delta(i,j)] \tag{C.7}$$

with :

$$\Delta(i,j) = h_{jj} - [h_{ii}h_{jj} - hij^2] - h_{ii} \tag{C.8}$$

where $h_{ij}$ is the element $ij$ of the Hat Matrix $H = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$ :

$$h_{ij} = \mathbf{x}_i'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{x}_j \tag{C.9}$$

Various termination criteria may be considered. For instance, the algorithm may be stopped when the increase of the determinant is smaller than a chosen value. As usual with greedy algorithms, local optima exist in general, so that the solution thus obtained may be sub-optimal.

## C.3 The Active Learning background

In contrast to classical learning (passive learning), the active learner selects the most useful experiments to be added to the initial data set. The learner chooses the best instances from a given set

of unlabeled examples (*pool-based sample selection* [Cohn 94] and [Melville & Mooney 04]).

The Active Learning strategy can be summarized by three steps :
- Train the learner using the current training set.
- Choose a point $\mathbf{x}$ in the pool of candidate experiments.
- Measure or compute the corresponding quantity of interest y and add the point ($\mathbf{x}$,y) to the training set.

This procedure is an incremental strategy, which adds new training points iteratively.

The main question in active learning is how to choose the point $\mathbf{x}$ in the second step. Various strategies may be considered, such as :
- adding experiments where data is missing,
- adding experiments where confidence in model predictions is low [Thrun & Möller 92],
- adding experiments in order to minimize the generalization error of the model, see for instance [Schohn & Cohn 00] for Support Vector Machine, or [Sung & Niyogi 92], where the *expected integrated squared difference* (which is an estimation of the generalization error in a bayesian framework) is minimized.

## C.4 Design of experiments by LDR-Bagging

We describe, in this section, new method called LDR for Learner Disagreement from experiment Resampling. As explained in section C.2.1, we denote by $\boldsymbol{\theta}_{\mathcal{L}}$ the vector of parameters for which the cost function is minimum :

$$f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}}) = \texttt{ArgMin}_{f(\mathbf{x},\boldsymbol{\theta})} \sum_{(y_k,\mathbf{x}_k)\in\mathcal{L}} l(y_k, f(\mathbf{x}_k, \boldsymbol{\theta})) \tag{C.10}$$

Clearly, different optimal parameter vectors will be derived from different training sets ; that variability can be investigated by resampling methods such as bagging (bootstrap aggregation) [Breiman 96] ; we first describe that method, which is central to our experimental planning technique.

### C.4.1 Bagging

Given a training set $\mathcal{L}$, the aggregated predictor is defined by :

$$f_A(\mathbf{x}) = E_{\mathcal{L}}[f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}})] \tag{C.11}$$

where $E_{\mathcal{L}}$ is the expectation value of the predictions of the model, for variable vector $\mathbf{x}$, for all possible training sets $\mathcal{L}$ of identical size ; the expectation value is estimated by the average, hence the subscript $A$.

The prediction provided by the aggregated predictor is more accurate than the average of the predictions provided by the individual predictors of the same family on the same data set :

$$\|f_A(\mathbf{x}) - y(\mathbf{x})\|^2 = \|E_{\mathcal{L}}[f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}})] - y(\mathbf{x})\|^2 \quad \text{(C.12)}$$

$$\leq E_{\mathcal{L}}\|f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}}) - y(\mathbf{x})\|^2 \quad \text{(C.13)}$$

An estimation of $f_A$ can conveniently be obtained by the bootstrap [Efron & Tibshirani 93], a statistical resampling method : examples are drawn randomly with replacement from the original data set $\mathcal{L}_n$, of size $n$, thereby generating an ensemble of $B$ data sets of identical size $n$. Denoting by $\mathcal{L}_n^{*b}$ the bootstrap sample (or replicate) $b$, the estimated expectation by bootstrap (hence the subscript $\mathcal{B}$) with $B$ replicates is :

$$f_{n,\mathcal{B}}(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}_n^{*b}}) \quad \text{(C.14)}$$

### C.4.2 Active Learning by LDR-Bagging

Similarly to estimating the expectation value of the predictions, their variance can be estimated by bootstrapping of the original data set :

$$\sigma^2_{\mathcal{B}}[f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}_n})] = \frac{1}{B-1}\sum_{b=1}^{B}(f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}_n^{*b}}) - f_{n,\mathcal{B}}(\mathbf{x}))^2 \quad \text{(C.15)}$$

The approach to active learning, or experimental planning, that we advocate here consists in adding new experiments in the regions of variable space where the bootstrap estimate of the variance of the predictions is largest, i.e. where the predictors constructed from data sets obtained by Bootstrap resampling disagree most. That can be viewed as a paradigm of a teacher-classroom interaction, where each student learns from a part of the data, the teacher asks questions, the classroom provides answers, and new questions are asked in the area where the greatest disagreement between all possible answers arises.
Therefore, our method can be summarized as follows :

– find the point of maximal prediction variance

$$\mathbf{x}_{\texttt{new}} = \texttt{ArgMax}_{x \in U}\sigma^2_B[f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}_n})] \quad \text{(C.16)}$$

– perform a numerical experiment, i.e. compute $f(\mathbf{x}_{\texttt{new}})$, and include the experiment $(\mathbf{x}_{\texttt{new}}, f(\mathbf{x}_{\texttt{new}}))$ in the initial sample $\mathcal{L}_n$ in order to obtain the new sample $\mathcal{L}_{n+1}$.

$$\mathcal{L}_{n+1} = \mathcal{L}_n \cup \{(\mathbf{x}_{\texttt{new}}, y(\mathbf{x}_{\texttt{new}}))\} \quad \text{(C.17)}$$

This active strategy is terminated when the decrease of the prediction variance is not significant, or when the predefined maximum number of additional experiments is reached (see the LDR-Algorithm Table C.1).

The generation of the bootstrap aggregated predictor (step 2) involves the training of the model by an appropriate procedure. For linear-in-their-parameters models, the procedure may be ordinary least squares ; for non-linear-in-their-parameters models, such as neural networks, training is performed by minimizing the chosen cost function. In the latter case, the main source of variability should be the resampling process, rather than the existence of local minima of the cost function ; to that end, for each bootstrap sample, several models are trained, and a single model is selected, as explained in section 6.1.1.

**LDR-Algorithm**

**Given :**
$T_n$ - set of training examples
$P$ - set of candidate experiments (generally a grid of variable space)
$y$ - unknown function
$k$ - number of selected candidate experiments, appended to the training set after selection
$n$ - size of each sample ($n \neq 0$)

Repeat $k$ times :

1. Generate $B$ bootstrap samples $\mathcal{L}_n^{*b}$
2. Generate the bootstrap aggregated predictor $f_{n,\mathcal{B}}(\mathbf{x})$
3. $\forall x_j \in P$ compute the estimated variance of the predictions $\sigma^2_{\mathcal{B}}[f(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{L}_n})]$
4. Select the point of maximal prediction variance in $P$, noted $\mathbf{x}_{\texttt{new}}$
5. Remove $\mathbf{x}_{\texttt{new}}$ from $P$ and add $(\mathbf{x}_{\texttt{new}}, y(\mathbf{x}_{\texttt{new}}))$ to $T_n$ ; $(n \leftarrow n+1)$.

TAB. C.1 – LDR algorithm

### C.4.3 A simple didactic example

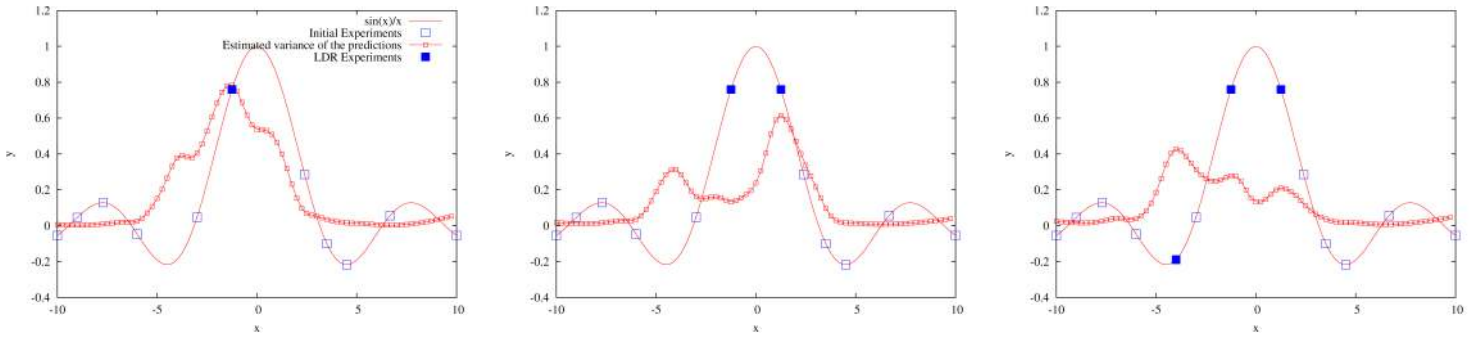
We illustrate the above procedure by the simple example of $sin(x)/x$. The initial training set features ten experiments, not uniformly distributed in variable space. The bootstrap estimates of the prediction variance over all candidate points of a grid are computed, and the point with maximum prediction variance estimate is included in the initial training set, as shown on Figure C.2.

The estimated prediction variance decreases significantly at each step in the vicinity of the new points, and decreases globally in the domain. The active strategy is terminated when the decrease of the predictive variance is not significant (we would have stopped the heuristic at the fifth step). Figure C.3 compares the generalization error of models that learned on data selected by LDR and on data sets generated randomly. The generalization error is estimated by an integration Monte Carlo method which provides an estimate of :

$$\int_U (y(\mathbf{x}) - f(\mathbf{x}, \boldsymbol{\theta}))^2 p(\mathbf{x}) d\mathbf{x} \tag{C.18}$$

It shows that the generalization error in the LDR case decreases significantly with the number of new experiments.

Note that we compare the generalization performance of LDR-designed models with the generalization performance of models obtained by training from a single random data set. Section C.6 reports comparisons between LDR designed models, D-optimality designed models, and models trained from 500 different random data sets.




(a) 1st iteration (b) 2nd iteration (c) 3rd iteration

(d) 4th iteration (e) 5th iteration (f) 6th iteration

FIG. C.2 – Active Learning with LDR-Bagging

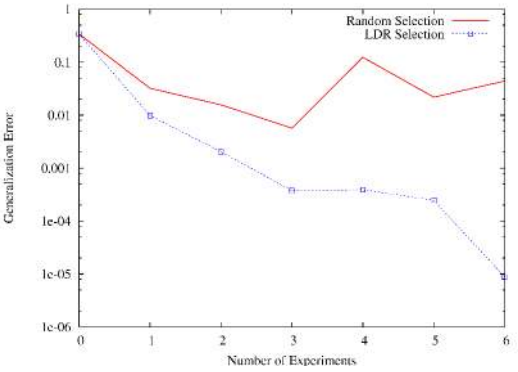



FIG. C.3 – Comparison of generalization error between models that learned on LDR-generated data sets and on random data sets.

## C.5 Active Learning by LDR-Leave One Out

The prediction variance may also be estimated by Leave-One-Out, especially when the models are linear in their parameters : in that case, the variance of the parameters can be computed explicitly. The application of that technique to a benchmark [SALTELLI & HOMMA 92] is described in section C.6.
The models were sought within the family of linear combinations of functions $\Psi_k$ resulting from the tensorisation of orthogonal Legendre polynomials. The family of orthogonal Legendre polynomials provides a well-conditioned information matrix.

$$y(\mathbf{x}) = \sum_{k=0}^{P} \theta_k \Psi_k(\mathbf{x}) = \Psi'(\mathbf{x})\boldsymbol{\theta} \tag{C.19}$$

The parameters $\theta_k$ are computed by ordinary least squares, as described in 1.1. or by SVD (Singular Value Decomposition). We denote by $\boldsymbol{\theta}_{(i)}$ the parameter vector obtained by removing example $i$ from the training set, by $h_{ii}$ the leverage of observation $i$ (the $i$-th diagonal element of the Hat Matrix $H = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$), and by $\epsilon_i$ the residual of example $i$ when it is present in the training set. The parameter vector $\boldsymbol{\theta}_{(i)}$ is obtained explicitly by [RAO & TOUTENBURG 95] :

$$\boldsymbol{\theta} - \boldsymbol{\theta}_{(i)} = (\mathbf{X}'\mathbf{X})^{-1}\frac{\mathbf{x}_i\epsilon_i}{1-h_{ii}} \tag{C.20}$$

We denote by $\alpha_i$ the quantity $\epsilon_i/(1-h_{ii})$, by $\Lambda$ the matrix of elements $\Lambda_{ij} = \alpha_i^2\delta_{ij} - \frac{1}{n}\alpha_i\alpha_j$ and by $\mathbf{X}^+ = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$ the pseudo-inverse of $\mathbf{X}$. The estimated prediction variance $\sigma_y^2(\mathbf{x})$ is given by :

$$\sigma_\theta^2 = \frac{1}{n}\mathbf{X}^+\Lambda\mathbf{X}^{+'} \Rightarrow \sigma_y^2(\mathbf{x}) = \frac{1}{n}\Psi'(\mathbf{x})\mathbf{X}^+\Lambda\mathbf{X}^{+'}\Psi(\mathbf{x}) \tag{C.21}$$

Since the prediction variance can be computed exactly (within numerical roundoff errors) from the pseudo-inverse $\mathbf{X}^+$, the new point $\mathbf{x}_{\mathtt{new}}$ can be obtained without resorting to resampling :

$$\mathbf{x}_{\mathtt{new}} = \mathtt{ArgMax}_{\mathbf{x}\in U}\|\Psi'(\mathbf{x})\mathbf{X}^+\Lambda^{1/2}\| \tag{C.22}$$

The new point is chosen in a set of candidate experiments.

## C.6 Results

In this section, the efficiencies of D-optimality, LDR active learning and random sampling of variable space are compared on three different problems.

### C.6.1 The Homma & Saltelli benchmark [Saltelli & Homma 92]

The method was validated on the Homma-Saltelli benchmark. The data-generating function is :

$$y(\mathbf{x}) = sin(x_1) + 7sin^2(x_2) + 0.1x_3^4 sin(x_1) \tag{C.23}$$

$$\text{with } x_i \in [-\pi; \pi] \forall i = 1, ..., 3$$

100 experiments were obtained by LHS, thereby generating the initial data set.

In the following, the models are feedforward neural networks (MLPs) with a single layer of hidden neurons. With the initial data set of 100 examples, we trained several MLPs with different numbers of hidden neurons. The generalization error of each MLP was estimated by the Monte Carlo integration method mentioned in section C.4.3. Models with twelve hidden neurons gave a good bias-variance tradeoff. The purpose of experimental planning was to supplement the initial training set of 100 examples with 60 additional examples. The generalization error was also estimated by the Monte Carlo integration method.

#### Results for LDR-Bagging method

**A comparison between D-optimality, LDR Active Learning and random sampling of variable space**

We compared D-optimality, LDR active learning and random sampling of variable space in order to estimate the accuracy of the first two planning techniques with respect to the accuracy of a random strategy. Since the random strategy is not representative with only one data set, 500 random data sets were generated, in order to provide a robust statistic of the random strategy accuracy.
Since neural network training depends on initial weights, we trained the MLP 50 times with different random weight initialization, allowing for a maximum number of cycles for each training. After these 50 trainings, we selected the MLP that had the smallest training error[20]. That time will be further reduced in the future by application of the method described in [25], based on the correlation between the MLP performance at convergence and its performance early in the training process, which allows discarding a model even before its training has been completed.
Figure C.4 shows the histogram of the estimated generalization errors of the 500 neural networks. The two lines are the estimated generalization errors of neural networks that learned on the D-optimality and the LDR-Bagging data sets.

As expected, both experimental design techniques led to better results than a random selection of experiments : models built on D-optimal training sets outperformed the random strategy in 86% of the cases, and the LDR method outperformed random selection in 91% of the cases.

[20]The mean square error on the training set is correlated with the generalization error when no measurement uncertainty is present.

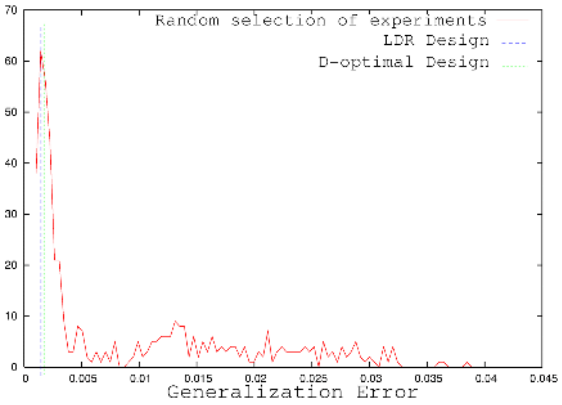



FIG. C.4 – Comparison between D-optimal design, LDR design and random strategy

## Comparison between D-optimality and LDR Active Learning

Since the cost function of neural networks has local minima, a statistical comparison between two experimental planning methods requires training the network with different initial values of the parameters. 1000 different neural networks were trained on each data set. Each neural network was selected on its training mean square error among 50 different neural networks. Figure C.5 shows the distribution of the generalization error estimates.

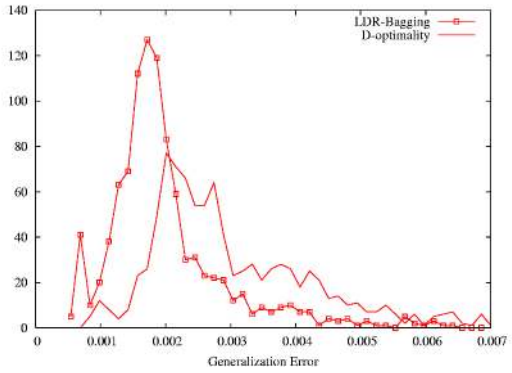
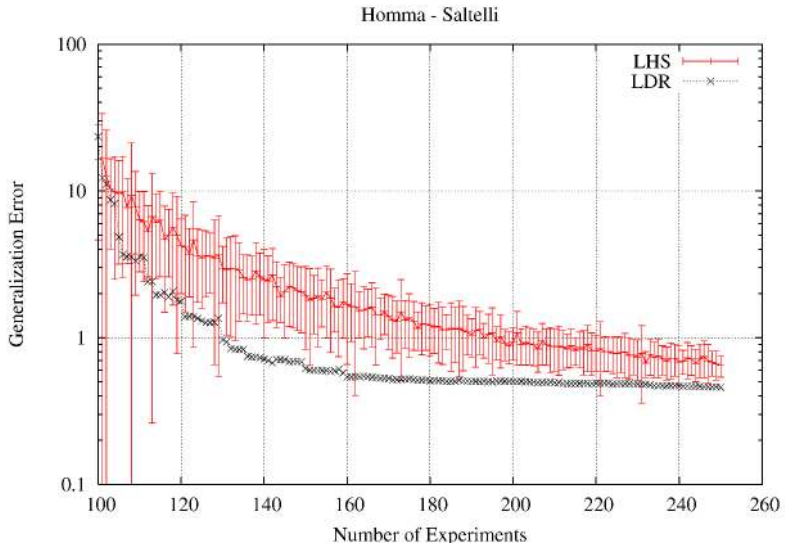



FIG. C.5 – Generalization error distribution of 1000 MLPs trained on D-optimal and LDR data sets.

For the LDR method, the average generalization error is $\mu_b = 3\mathbf{x}10^{-3}$. For D-optimal design, the average is $\mu_d = 5.7\mathbf{x}10^{-3}$.
The models that learned on LDR samples appear to be more efficient than the D-optimal ones. Indeed, 73% of the LDR models have a generalization error smaller than $2\mathbf{x}10^{-3}$ against 26% in the D-optimal case.

### Results for LDR-Leave One Out Method

We compared the generalization error, estimated by Monte Carlo, of models constructed on several samples of the same size (100 to 250 experiments), generated by LDR-Leave-one-out and by LHS. In order to obtain a robust comparison, we used 50 LHS samples for each sample size,




(a)

FIG. C.6 – Comparison between LHS and LDR selection. To approximate the Homma-Saltelli function, the models were sought as linear combinations of Legendre polynomials of degree 6. The prediction variance was estimated by (virtual) leave-one-out. For LHS strategy, in every iteration $k = 1, ..., 150$, 50 LHS samples of size $100 + k$ were generated. The mean and the standard deviation are shown. For LDR strategy, a single point is added at each iteration. The selected point is the point for which the prediction variance is maximum.

and we computed the average and the standard deviation of the generalization error.

Figure C.6 shows the average evolution of the generalization error, and its standard deviation, of models that learned on both LHS and LDR samples. In real applications, the initial samples (100 experiments) would be based on low-discrepancy mathematical series, which are more robust, on the average, than the LHS samples [26].
In that case, the LDR-Leave one out active learning appears to be more efficient than LHS. For samples of identical size, the generalization error of models that learned on LDR samples is smaller than the average generalization error of LHS by at least the standard deviation of LHS generalization error.

### C.6.2 The Friedman benchmark

In that case, the data-generating function is the Friedman function :

$$y(\mathbf{x}) = \theta_1 sin(\pi x_1 x_2) + \theta_2 (x_3 - \theta_3)^2 + \theta_4 x_4 + \theta_5 x_5 \tag{C.24}$$

$$\text{with } \theta_1 = \theta_4 = 0.4,\ \theta_2 = 0.8,\ \theta_3 = 0.5,\ \theta_5 = 0.2 \text{ and } x_i \in [0; 1] \forall i = 1, ..., 5$$

An initial data set of 100 experiments was generated by LHS.

For this benchmark, we used the same test procedure used in Homma & Saltelli benchmark. In the following, the models are feedforward neural networks (MLPs) with a single layer of six

hidden neurons [21]. The purpose of experimental planning was to supplement the initial training set of 100 examples with 30 additional examples. The generalization error was estimated by the Monte Carlo integration method C.4.3.

### Results for LDR-Bagging method

#### A comparison between D-optimality, LDR Active Learning and random sampling of variable space

We estimated the accuracy of the D-optimality and the LDR active learning planning techniques with respect to the accuracy of a random strategy with 500 random data sets. In each case, we used an accurate neural network selected on its training mean square error among 50 different neural networks.

Figure C.7 shows the histogram of the estimated generalization errors of the 500 neural networks. The two lines are the estimated generalization errors of neural networks that learned on the D-optimality and the LDR-Bagging data sets.

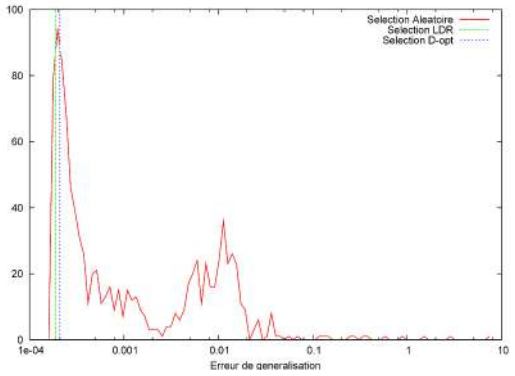



FIG. C.7 – Comparison between D-optimal design, LDR design and random strategy.

As expected, both experimental design techniques led to better results than a random selection of experiments : models built on D-optimal training sets outperformed the random strategy in 94.8% of the cases, and the LDR method outperformed random selection in 96.7% of the cases.

#### Comparison between D-optimality and LDR Active Learning

We performed a statistical comparison between D-optimality and LDR active learning. 1000 different neural networks were trained on each data set. Each neural network was selected on its training mean square error among 50 different neural networks. Figure C.8 shows the distribution of the generalization error estimates. In that case, both experimental design techniques have the same accuracy.

[21] Models with six hidden neurons gave a good bias-variance tradeoff.

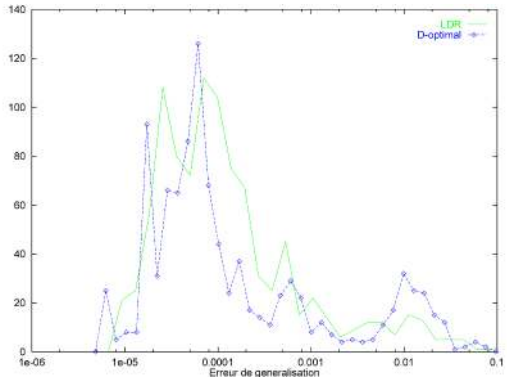



FIG. C.8 – Generalization error distribution of 1000 MLPs trained on D-optimal and LDR data sets.

### C.6.3 The CEA application

For this application, we used a data set of more than 2000 real examples generated by a simulation model of a physical process. The quantity to be predicted is the multi-keV x-ray conversion efficiencies in the context of multi-keV x-ray production from prepulsed germanium foils.
An initial data set of 35 experiments was generated by LHS. For this benchmark, we used the same test procedure used in Homma & Saltelli and Friedman benchmarks.
In the following, the models are feedforward neural networks (MLPs) with three variables and a single layer of six hidden neurons. The purpose of experimental planning was to supplement the initial training set of 35 examples with 36 additional examples. The generalization error was estimated by the Monte Carlo integration method C.4.3.

#### Results for LDR-Bagging method

**A comparison between D-optimality, LDR Active Learning and random sampling of variable space**

In the previous cases, the efficiency of D-optimal planning and of LDR active learning were compared to the efficiency of a random strategy with 500 random data sets. In each case, we used an accurate neural network selected on its training mean square error among 50 different neural networks.
Figure C.9 shows the histogram of the estimated generalization errors of the 500 neural networks. The two lines are the estimated generalization errors of neural networks that learned on the D-optimality and the LDR-Bagging data sets.
As expected, both experimental design techniques led to better results than a random selection of experiments : models built on D-optimal and LDR training sets outperformed the random strategy.

**Comparison between D-optimality and LDR Active Learning**

We performed a statistical comparison between D-optimality and LDR active learning. 1000 different neural networks were trained on each data set. Each neural network was selected on its training mean square error among 50 different neural networks. Figure C.10 shows the distribution of the generalization error estimates.

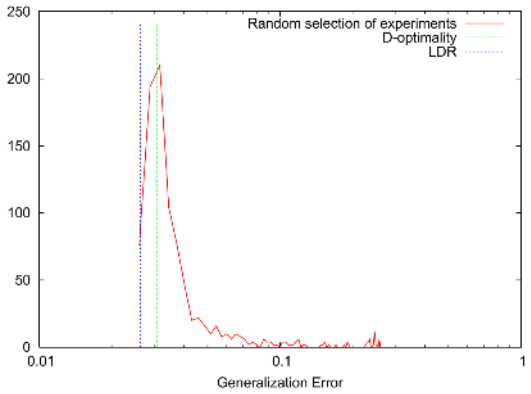



FIG. C.9 – Comparison between D-optimal design, LDR design and random strategy.

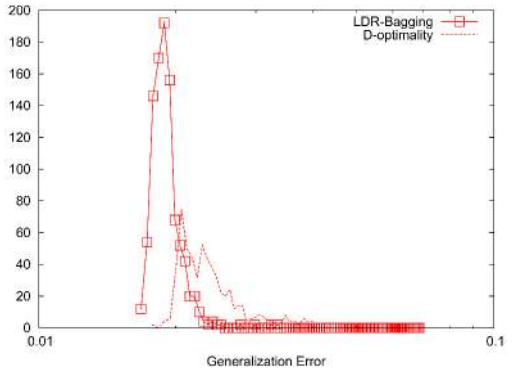



FIG. C.10 – Generalization error distribution of 1000 MLPs trained on D-optimal and LDR data sets.

The models that learned on LDR samples appear to be more efficient than the D-optimal ones. For finding the point of maximal prediction variance, the computational burden is the following : train 50 MLPs for selection and train 200 MLPs for computing the prediction variance with 200 replicates. The planning of an experiment takes a few minutes on a present-day PC, which is a negligibe overhead with respect to the time necessary for performing the numerical experiment iself.

## C.7 Conclusion

A new active learning strategy (LDR for Learner Disagreement from experiment Resampling), intended for use in the context of the planning of numerical experiments, has been described. The traditional optimal methods for experimental design give optimum data sets by minimizing the variability of the parameters due to experimental noise. In a context of numerical experiments, no experimental noise is present, so that the traditional approaches are not relevant. In order to generate a data set, the LDR method estimates the variance of the prediction of several models around the bagged predictor, and plans a new experiment at the location, in the space of variables, where the estimated prediction variance is maximal. The procedure is somewhat computer-intensive, since it is based on resampling, but the computation time necessary for planning an experiment is negligibly small as compared to the computation time required by the experiment itself. A comparison between the prediction errors of models that learned on data sets designed by LDR and D-optimal design leads to the conclusion that the LDR method gives promising results in terms of quality of models that learned on such designs.

## Acknowledgment

This work was performed in part within the framework of the NeuroPex project on experimental planning for neural network models. Among the partners were the Netral company, the CEA (DAM/DP2I) and the CEA (DEN/DM2S). This work was supported in part by a CEA grant.

# Bibliographie


[1] V. V. Fedorov, "Theory of Optimal Experiments", Academic Press, New York

[2] J. -P. Gauchi, "Plans d'expériences optimaux pour modèles linéaires", Plans d'expériences - Applications à l'entreprise, ch. 7, Editions Technip

[3] J. Kiefer, "Optimum Experimental Designs", *JR Statist. Soc.*, vol. 21, pp. 272-319, 1959

[4] J. Kiefer and J. Wolfowitz, "Optimum Designs in Regression Problems", *Ann of Math Statist*, vol. 30, pp. 271-294, 1959

[5] H. P. Wynn, "The Sequential Generation of D-Optimum Experimental Designs", *Ann of Math Statist*, vol. 41, pp. 1655-1664, 1970

[6] J. -P. Vila, "Local optimality of replications from a minimal D-optimal design in regression : a sufficient and a quasi-necessary condition", *Journal of Statistics Planning and Inference*, vol. 29, pp. 261-277, 1991

[7] D. MacKay, "Information-based Objective Fuctions for Active Data Selection", *Neural Computation*, vol.4, 1992

[8] D. Cohn, "Neural Networks Exploration Using Optimal Experiment Design", *Advances in Neural Information Processing Systems*, vol. 6, 1994

[9] S. Issanchou and J. -P. Gauchi, "Plans d'expériences optimaux pour réseaux de neurones", presented at ChimioMetrie 2004, Paris, France, 2004

[10] M. Witczak, "Toward the Training of Feed-Forward Neural Networks with the D-Optimum Input Sequence", *IEEE Transactions on Neural Networks*, vol. 17, no. 2, 2006

[11] C. R. Rao and H. Toutenburg, "Linear Models - Least squares and alternatives", ch. 7, Springer Series in Statistics

[12] T. J. Mitchell, "An algorithm for the construction of D-optimal experimental designs", *Technometrics*, vol. 16, pp. 203-210, 1974

[13] D. M. Bates and D. G. Watts, "Nonlinear regression analysis and its applications", New York : Wiley, 1988

[14] G. Monari and G. Dreyfus, "Local Overfiting control via leverages", *Neural Computation*, vol. 14, 2002

[15] R. L. Iman and J. C. Helton and J. E. Campbell, "An approach to sensitivity analysis of computer models, Part I. Introduction, input variable selection and preliminary variable assessment", *The Journal of Quality Technology*, vol. 13, pp. 174-183

[16] A. C. Atkinson and A. N. Donev, "The construction of exact D-optimal designs with application in blocking response surface designs", *Biometrika*, 76, 1989

[17] D. Cohn and L. Atlas and R. Ladner, "Improving generalization with active learning", *Machine Learning*, vol. 15, pp. 201-221, 1994

[18] P. Melville and R. Mooney, "Diverse ensembles for Active Learning", in *Proceedings of 21st Internationnal Conference on Machine Learning*, Banff, Canada, 2004

[19] S. Thrun and K. Möller, "Active Exploration in Dynamic Environnements", *Advances in Neural Information Processing Systems*, vol. 4, Morgan Kaufmann, 1992

[20] G. Schohn and D. Cohn, "Less is More : Active Learning with Support Vector Machines", in *Proceedings of 17th International Conference on Machine Learning*, pp. 839-846, San Francisco, CA Morgan Kaufmann, 2000

[21] K. K. Sung and P. Niyogi, "Active Learning for Function Approximation", *Advances in Neural Information Processing Systems*, vol. 7, 1995

[22] L. Breiman, "Bagging Predictors", *Machine Learning*, vol. 24, no. 2, pp. 123-140, 1996

[23] B. Efron and R. Tibshirani, "An introduction to the Bootstrap", Chapman & Hall, 1993

[24] A. Saltelli and T. Homma, "Sensivity Analysis for Model Output, Performances of Black Box Techniques on Three International Benchmark exercises", *Computational Statistics & Data Analysis*, vol. 13, pp. 73-94, 1992

[25] L. A. Feldkamp and D. V. Prokhorov and C. F. Eagen, "Multiple-Start Directed Search for Improved NN Solution", *Proceedings IJCNN 2004*, pp 991-996, Budapest Hungary, July 2004

[26] H. Niederreiter, "Random number generation and quasi-monte-carlo methods", *Society of Industrial and Applied Mathematics*, 1992

# Index

Index

A-optimalité, 93
Algorithme
  d'échange double de Fedorov, 125
  de Levenberg-Marquardt, 175
Apprenti, 30

Bagging, 139
Biais, 33
Bootstrap, 55, 96
  analyse combinatoire, 58
  en régression, 58
  variance des prédictions, 65

Complexité
  choix d'une, 98
Consistance d'un apprentissage, 31
Critère
  de A-optimalité, 121
  de D-optimalité, 122
  de E-optimalité, 121
  de G-optimalité, 123
  de J-optimalité, 123
  d'Akaike, 93
  de D-optimalité locale, 130

Dilemme Biais-Variance, 33, 37
Dimension VC, 31

Early stopping, 38, 105
Erreur
  d'approximation, 88
  d'estimation, 88
  de généralisation, 31
    estimation, 87

Fonction
  d'erreur de prédiction, 119
  de variance de prédiction, 119

Générateur, 30

Hyperellipsoïde de confiance, 120

Initialisation des paramètres, 105

LDR, 139
Leave-Many-Out, 62
Leave-One-Out, 45, 94
  virtuel
    pour modèles linéaires, 95
    pour modèles non linéaires, 95
Leave-One-Out Virtuel, 46
Levier, 46
Leviers
  Calcul des, 54
  Interprétation des, 50
LHS, 77

Matrice
  chapeau, 47
  d'information, 35
  de variance-covariance, 117
  des effets, 35
  du modèle, 35, 117

Neurone
  caché, 36
  de sortie, 36

Plan d'expériences, 115

Régresseur, 34
Régularisation, 37
Répliques Bootstrap, 56
Réseaux de neurones, 36
Rétropropagation du gradient, 169, 170
Risque
  empirique, 31
  fonctionnel, 30
  structurel, 33
  théorique, 30

Solution des moindres carrés, 35
Sous-paramétrisation, 33
Superviseur, 30
Sur-ajustement, 37
Sur-paramétrisation, 33

Validation croisée, 94
Variable
    primaire, 34
    secondaire, 34
Variance, 33
Variance des prédictions, 65

Weight decay, 38

X-optimalité, 131

# Table des figures


1.1 Schéma de l'apprentissage supervisé . . . . . 30
1.2 Compromis entre le risque empirique et le risque fonctionnel . . . . . 33
1.3 Représentation graphique d'un réseau de neurones. . . . . 36
1.4 Evolution de l'erreur de généralisation en fonction du nombre de cycles d'apprentissage. . . . . 38

2.1 Mesures effectuées sur un processus à une variable . . . . . 51
2.2 Leviers des points correspondant au mesures du processus de la figure 2.1 . . . . . 51
2.3 Répartition de 6 points . . . . . 53
2.4 Les leviers se situent sur une parabole qui atteint son minimum en $x = 1$ qui est la moyenne des $x$ 53
2.5 Schéma du bootstrap établissant une statistique d'un estimateur $\mu$ . . . . . 56
2.6 Probabilité qu'un individu n'apparaisse pas dans une réplique pour différentes valeurs du cardinal de la réplique $N$. . . . . 57
2.7 Utilisation de Bootstrap en régression . . . . . 59
2.8 Probabilité $P(k/N)$ d'avoir une fraction $k/N$ d'exemples différents (calculée pour $N = 100$). En abscisse la valeur de la fraction $k/N$, en ordonnée le logarithme de $P_N(k/N)$ . . . . . 60
2.9 Bootstrap : probabilité d'avoir $k$ exemples distincts pour différentes valeurs de la taille de l'échantillon initial : $N = 20, 25, 50, 75, 100, 150$ . . . . . 61
2.10 Bootstrap : fonction de répartition du taux $k/N$ d'exemples distincts pour différentes valeurs de la taille de l'échantillon initial : $N = 25, 50, 75, 100, 150, 200$. Il faut garder à l'esprit que $k/N$ n'est pas une variable continue et ne peut prendre que $N$ valeurs entre 0 et 1. . . . . 61
2.11 Points de la base d'apprentissage de la fonction sinus cardinal et le modèle obtenu. . . . . 65
2.12 Modèles obtenus à partir des 200 répliques bootstrap dans l'espace normalisé . . . . . 66
2.13 Modèles obtenus à partir des 200 répliques bootstrap dans l'espace de départ . . . . . 67
2.14 Corrélation entre la variance des prédictions par Bootstrap et les leviers pour un processus déterministe . . . . . 67
2.15 Répartition de la variance bootstrap des prédictions pour $covar_{\mathcal{B}}(\theta_0, \theta_1) < 0$. . . . . 68
2.16 200 modèles construits à partir de répliques bootstrap. On retrouve le minimum de la variance des estimations localisé en $x \simeq 6$ par la Figure 2.15. . . . . 69
2.17 Relation entre la variance bootstrap des prédictions et les leviers pour un modèle déterministe pour lequel on estime le terme constant. . . . . 70
2.18 Relation entre la variance bootstrap des prédictions et les leviers pour différentes valeurs de $\mathrm{cov}(\theta_0,\theta_1)$ . . . . . 71
2.19 La surface sinus cardinal. . . . . 72
2.20 Lignes de niveau de la surface décrite par les leviers. . . . . 72
2.21 Lignes de niveau de la surface décrite par les leviers et celle de la variance bootstrap. . . . . 73
2.22 Relation entre la variance bootstrap et les leviers pour un modèle linéaire en dimension 2. . . . 73

2.23 Lignes de niveau de la surface décrite par les leviers et celle de la variance bootstrap dans le cas d'un modèle affine. . . . . 74
2.24 Le sinus cardinal centré. . . . . 75
2.25 Projection sur le même plan des leviers et de la variance bootstrap. . . . . 75
2.26 Corrélation entre la variance bootstrap et les leviers pour le sinus cardinal centré. . . . . 76
2.27 Tirage LHS pour lequel chacune des variables est tirée suivant une loi uniforme. . . . . 78
2.28 Tirage LHS pour lequel la variable $x_1$ est tirée suivant une loi normale et la variable $x_2$ tirée suivant une loi uniforme. Chaque cellule de la grille présente la même probabilité de tirage. . . 78
2.29 La surface polynomiale de degré 3 approchant le sinus cardinal. . . . . 79
2.30 Les leviers pour la surface polynomiale de degré 3. . . . . 80
2.31 Lignes de niveau des leviers pour la surface polynomiale de degré 3. . . . . 80
2.32 La variance bootstrap pour la surface polynomiale de degré 3. . . . . 81
2.33 Lignes de niveau de la variance bootstrap pour la surface polynomiale de degré 3. . . . . 81
2.34 Surface décrite par les éléments diagonaux de la matrice $H_W$ pour la surface polynomiale de degré 3. . . . . 82
2.35 Lignes de niveau des éléments diagonaux de la matrice $H_W$ pour la surface polynomiale de degré 3. 83

3.1 Pourcentage de répliques pour lesquelles le taux d'exemples différents est inférieur à $X$ . . . . 100
3.2 Base d'exemples de la fonction sinus cardinal . . . . . 101
3.3 Erreurs de généralisation (EQMT) estimées par bootstrap, par validation croisée, et à partir d'un maillage de 2000 points en fonction de la complexité de la famille de fonctions ; il s'agit ici de polynômes : l'axe des abscisses représente donc le degré du polynome. . . . . 102
3.4 Réponse du modèle polynomial de degré 19 construit à partir de la base LHS du sinus cardinal. 102
3.5 Erreurs de généralisation estimées par bootstrap, par validation croisée et à partir d'un maillage de 2000 points en fonction du degré du polynôme. . . . . 103
3.6 La fonction $sin(x)cos(0.25x)$ sur l'intervalle $[-4; 10]$ . . . . . 103
3.7 Erreurs de généralisation estimées par bootstrap, par validation croisée et à partir d'un maillage de 2000 points en fonction du degré du polynôme. . . . . 104
3.8 La fonction $x^{sin(x)}$ sur l'intervalle $[0; 16]$ . . . . . 104
3.9 Erreurs de généralisation estimées par bootstrap et par validation croisée et erreur de généralisation obtenue à partir d'un maillage de 2000 points en fonction du degré du polynome. . . . . 105
3.10 Corrélations entre l'erreur quadratique moyenne d'apprentissage et l'erreur quadratique moyenne de test dans le cadre d'un processus déterministe et pour des réseaux de neurones de même complexité. . . . . 107
3.11 Corrélation entre l'erreur quadratique moyenne d'apprentissage et l'erreur quadratique moyenne de test dans le cadre du processus déterministe de Homma et Saltelli pour une base comportant 100 exemples et pour des réseaux de neurones de même complexité. . . . . 107
3.12 Corrélations entre l'erreur quadratique moyenne d'apprentissage et l'erreur quadratique moyenne de test dans le cadre du processus déterministe de Homma et Saltelli pour une base comportant 200 exemples et pour des réseaux de neurones de même complexité. . . . . 108
3.13 Utilisation du bootstrap pour déterminer la statistique des sorties estimées par le modèle. Les modèles obtenus à partir des répliques bootstrap sont initialisés avec le même vecteur de paramètres qui correspond au vecteur de paramètres initial du modèle, créé à partir de la base complète, qui a obtenu l'erreur d'apprentissage la plus faible. . . . . 109
3.14 Le modèle correspondant aux pointillés est le meilleur en apprentissage parmi un grand nombre de modèles entrainés avec différentes initialisations. Son jeu de paramètres initial est utilisé pour initialiser tous les modèles dont la base d'apprentissage est une réplique bootstrap. . . . . 109

4.1 Domaine expérimental sans contraintes . . . 115
4.2 Domaine expérimental sous contraintes . . . 116
4.3 Zone de confiance des paramètres estimés . . . 121
4.4 Valeur du déterminant normé en fonction du nombre $N$ d'expériences du plan. . . . 129
4.5 Lien entre placement des points et variance des paramètres . . . 130

5.1 Schéma de l'apprentissage actif. . . . 137
5.2 Modèle agrégé obtenu à partir du ré-échantillonnage bootstrap. . . . 140
5.3 Apprentissage actif par la méthode LDR sur le sinus cardinal . . . 143
5.4 Comparaison de l'erreur de généralisation entre des modèles ayant appris sur les bases engendrée par LDR d'une part, et par planification aléatoire d'autre part. . . . 143
5.5 Répartitions des points LHS sur le domaine $[-4; 10]^2$ pour le sinus cardinal en dimension 2. . . . 144
5.6 Répartitions des points LHS sur la surface sinus cardinal. . . . 144
5.7 Profil de variance des prédictions estimée par Bootstrap pour le sinus cardinal en dimension 2. . . 145
5.8 Profils de variance pour différentes itérations de replanification pour le sinus cardinal en dimension 2. . . . 146
5.9 Répartition des points spécifiés par la méthode LDR parmi les points de la base d'apprentissage. 147
5.10 Histogramme des erreurs de généralisation de modèles dont les paramètres ont été estimés à partir de 1000 plans aléatoires. L'erreur de généralisation est évaluée sur un échantillon de 20 000 exemples. Les deux segments verticaux sont les erreurs de généralisation des modèles dont l'apprentissage a été conduit à partir des plans obtenus par la D-optimalité et la méthode LDR. 148
5.11 Histogramme des 1000 erreurs de généralisation pour la planification D-optimale et la planification LDR. Nous avons conduit l'apprentissage de 1000 modèles, chacun d'entre eux est le meilleur modèle en apprentissage parmi un ensemble de 30 modèles initialisés différemment. Nous avons calculé pour chacun d'eux l'erreur de généralisation sur un ensemble de test de 20 000 exemples. Cette figure représente l'histogramme des valeurs d'erreur de généralisation de ces 1000 modèles pour la D-optimalité et pour la méthode LDR. . . . 149
5.12 Fonction de répartition de l'erreur de généralisation pour la planification D-optimale et la planification LDR. . . . 150
5.13 Évolution de l'erreur de généralisation des modèles dont l'apprentissage a été effectué sur les bases planifiées par la méthode LDR et la D-optimalité, en fonction du nombre d'exemples que contiennent ces bases. . . . 150
5.14 Erreur d'apprentissage et erreur de test pour différentes architectures et pour différentes tailles de bases d'apprentissage. Chaque architecture est représentée par un code de couleur différent. . 152
5.15 Plan LHS initial vu en projection. . . . 153
5.16 Histogramme des 1000 erreurs de généralisation des modèles obtenus par apprentissage à partir de bases obtenues par planification aléatoire. Nous avons conduit l'apprentissage de modèles (le meilleur modèle en apprentissage parmi 30) sur chacune des bases aléatoires. Les deux segments verticaux sont les erreurs de généralisation des modèles dont l'apprentissage a été conduit à partir des plans obtenus par la D-optimalité et la méthode LDR. . . . 153
5.17 Histogramme des 1000 erreurs de généralisation pour la planification D-optimale et la planification LDR. Nous avons créé 1000 modèles (chacun d'eux étant le meilleur modèle en apprentissage parmi 30). Nous avons calculé l'erreur de généralisation sur chacun d'eux. Cette figure représente l'histogramme de ces 1000 valeurs pour la D-optimalité et pour la méthode LDR. . . . 154

5.18 Fonction de répartition de l'erreur de généralisation des modèles construits à partir des bases D-optimale et LDR. Certains modèles construits à partir du plan D-optimal ont obtenu une erreur de généralisation supérieure à $7.10^{-3}$ que nous n'avons pas reportée sur les graphiques. Ceci explique pourquoi la fonction de répartition pour la D-optimalité n'atteint pas 100 % pour une erreur de généralisation de $7.10^{-3}$. . . . . 155
5.19 Plan D-optimal final vu en projection. . . . . 155
5.20 Plan LDR final vu en projection. . . . . 155
5.21 Type de point replanifié pour les deux méthodes. . . . . 155
5.22 Histogramme des erreurs de généralisation (en echelle log) de 1000 modèles entrainés sur la base initiale LHS commune aux deux stratégies et les histogrammes des erreurs de généralisation de 1000 modèles entrainés sur une base LHS de 160 points et sur les bases D-optimale et LDR finales. 156
5.23 Comparaison entre les bases LHS et LDR. Nous avons approché la fonction de Homma et Saltelli par des polynômes de degré 6. La variance des prédictions est estimée par leave-one-out virtuel. Pour la stratégie LHS, pour toutes les itérations $k = 1, ..., 150$, 50 bases LHS de taille $100 + k$ ont été créées. Nous avons reporté la moyenne et l'écart-type de l'erreur de généralisation des modèles construits sur ces bases. Pour la méthode LDR, un seul point est ajouté à chaque itération. Le point choisi est celui pour lequel la variance des prédictions est maximale. . . . . 158

C.1 Confidence area for 2 parameters . . . . . 183
C.2 Active Learning with LDR-Bagging . . . . . 188
C.3 Comparison of generalization error between models that learned on LDR-generated data sets and on random data sets. . . . . 190
C.4 Comparison between D-optimal design, LDR design and random strategy . . . . . 192
C.5 Generalization error distribution of 1000 MLPs trained on D-optimal and LDR data sets. . . . 192
C.6 Comparison between LHS and LDR selection. To approximate the Homma-Saltelli function, the models were sought as linear combinations of Legendre polynomials of degree 6. The prediction variance was estimated by (virtual) leave-one-out. For LHS strategy, in every iteration $k = 1, ..., 150$, 50 LHS samples of size $100 + k$ were generated. The mean and the standard deviation are shown. For LDR strategy, a single point is added at each iteration. The selected point is the point for which the prediction variance is maximum. . . . . 193
C.7 Comparison between D-optimal design, LDR design and random strategy. . . . . 194
C.8 Generalization error distribution of 1000 MLPs trained on D-optimal and LDR data sets. . . . 195
C.9 Comparison between D-optimal design, LDR design and random strategy. . . . . 195
C.10 Generalization error distribution of 1000 MLPs trained on D-optimal and LDR data sets. . . . 196

# Bibliographie


[Abe & Mamitsuka 98] N. Abe & H. Mamitsuka,
*Query Learning Strategies using Boosting and Bagging*,
Proceedings of the Fifteenth International Conference on Machine Learning, pp 1-9, 1998
.

[Akaike 70] H. Akaike,
*Statistical predictor identification*,
Ann. Inst. Stat. Math., 22 :202-217, 1970
.

[Antoniadis *et al.* 92] A. Antoniadis, J. Berruyer & R. Carmona,
*Régression non linéaire et applications*,
Economica, 1992
.

[Atkinson & Donev 89] A. Atkinson & A. Donev,
*The construction of exact D-optimal designs with application in blocking response surface designs*,
Biometrika, 76, 1989
.

[Belsley *et al.* 80] D. Belsley, E. Kuh & R. Welsch,
*Regression Diagnostics*,
Wiley, 1980
.

[Bousquet & Elisseeff 02] O. Bousquet & A. Elisseeff,
*Stability and generalization*,
Machine Learning Research, 2, pp 499-526, 2002
.

[Box & Draper 59] G. Box & N. Draper,
*A basis for selection of a response surface design*,
Journal of the American Statistic Association, 54, 622-653, 1959
.

[Box & Draper 87] G. Box & N. Draper,
*Empirical model-building and response surfaces*,
Edition John Wiley and Sons, 1987
.

[Breiman 96] L. Breiman,
*Bagging Predictors*,
Machine Learning, 24, 2, 123-140, 1996
.

[Burges 98] C. Burges,
*A tutorial on support vector machines for pattern recognition*,
Data Mining and Knowledge Discovery 121-167, 1998
.

[Chapelle 04] O. Chapelle,
*Support Vector Machines : Principes d'induction, Réglage automatique et Connaissance a priori*, Thèse de doctorat,
Université Pierre et mMrie Curie - Paris VI, 2004
.

[Chernoff 53] H. Chernoff,
*Locally optimum designs for estimating parameters*,
Ann. of Math. Statist. 24, 586-602, 1953
.

[Cichocki & Unbehauen 93] A. Cichocki & R. Unbehauen,
*Neural Networks for Optimization and Signal Processing*,
Wiley, 1993
.

[Cohn 94] D. Cohn,
*Neural Networks Exploration using Optimal Experiment Design*,
Advances in Neural Information Processing Systems 6, 1994
.

[Cohn *et al.* 90] D. Cohn, L. Atlas & R. Ladner,
*Training connectionist networks with queries and selective sampling*,
Advances in Neural Information Processing Systems 2, Morgan Kaufmann, 1990
.

[Cohn *et al.* 94] D. Cohn, L. Atlas & R. Ladner,
*Improving generalization with active learning*,
Machine Learning, 15, 201-221, 1994
.

[Cover 65] T. Cover,
*Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition*,
IEEE Transaction on Electronic Computers 14 :326-334, 1965
.

[Dreyfus *et al.* 04] G. Dreyfus, J. Martinez, M. Samuelides, M. Gordon, F. Badran, S. Thiria & L. Hérault,
*Réseaux de neurones - Méthodologie et applications*,
Eyrolles, 2004
.

[Droesbeke *et al.* 97] J.-J. Droesbeke, J. Fine & G. Saporta,
*Plans d'expériences - Application à l'entreprise*,
Editions Technip, 1997
.

[Efron 79] B. Efron,
*Bootstrap methods :another look at the jackknife*,

Ann. Stat. 7, 1979
.

[Efron & Tibshirani 93] B. Efron & R. Tibshirani,
*An introduction to the Bootstrap*,
Chapman & Hall, 1993
.

[Fedorov 72] V. Fedorov,
*Theory of Optimal Experiments*,
Academic Press - New York, 1972
.

[Fisher 25] R. Fisher,
*Statistical methods for research workers*,
Olivier and Boyd, 1925
.

[Fisher 35] R. Fisher,
*The design of experiments*,
Olivier and Boyd, 1935
.

[Gauchi 97a] J.-P. Gauchi,
*Plans d'expériences optimaux pour modèles linéaires*,
Plan d'expériences - Application à l'entreprise - Editions Technip, 1997
.

[Gauchi 97b] J.-P. Gauchi,
*Plans d'expériences optimaux pour modèles non linéaires*,
Plan d'expériences - Application à l'entreprise - Editions Technip, 1997
.

[Gauchi 05] J.-P. Gauchi,
*Plans d'expériences optimaux : un exposé didactique*, vol. 33,
Revue Modulad, pages 139-162, 2005
.

[Gaudier 99] F. Gaudier,
*Modélisation par réseaux de neurones - Application à la gestion du combustible dans un réacteur*, Thèse de doctorat,
Ecole Normale Supérieure de Cachan, 1999
.

[Gazut & Martinez 04] S. Gazut & J.-M. Martinez,
*Apport de la théorie de l'apprentissage statistique à la construction de surfaces de réponse*,
Journées MAS de la SMAI - Contrôle stochastique et statistique, Nancy, 2004
.

[Gazut & Martinez 05] S. Gazut & J.-M. Martinez,
*Plans d'expériences itératifs à la construction de surfaces de réponse neuronales*,
MAMERN - Méthodes d'approximation, Oujda, Maroc, 2005
.

[Gazut *et al.* 07] S. Gazut, J.-M. Martinez, G. Dreyfus & Y. Oussar,
*Towards Optimal Design of Numerical Experiment*,

IEEE Transaction on Neural Networks, submitted, 2007
.

[Gazut *et al.* 06] S. Gazut, J.-M. Martinez & S. Issanchou,
*Plans d'expériences itératifs pour la construction de modèles non linéaires*,
38ièmes Journées de Statistique, Clamart, France, 2006
.

[Gilardi & Faraj 04] N. Gilardi & A. Faraj,
*Design of experiments by committee of neural networks*,
IEEE International Joint Conference on Neural Networks, 2004
.

[Goupy 99] J. Goupy,
*Plans d'expériences pour surfaces de réponse*,
DUNOD, 1999
.

[Grandvalet 00] Y. Grandvalet,
*Bagging down-weights leverage points*,
IJCNN, IV, 505-510, 2000
.

[Iman *et al.* 81] R. Iman, J. Helton & J. Campbell,
*An approach to sensitivity analysis of computer models, Part I. Introduction, input variable selection and preliminary variable assessment*,
The Journal of Quality Technology, vol.13, 1981
.

[Issanchou & Gauchi 04] S. Issanchou & J.-P. Gauchi,
*Plans d'expériences optimaux pour réseaux de neurones*,
ChimioMetrie, 2004
.

[Kiefer 59] J. Kiefer,
*Optimum experimental designs*,
J.R. Statist. Soc. B., 21, 272-319, 1959
.

[Kobilinsky 97] A. Kobilinsky,
*Les plans factoriels*,
Plan d'expériences - Application à l'entreprise - Editions Technip, 1997
.

[Levenberg 44] K. Levenberg,
*A method for the solution of certain non-linear problems in least squares*,
Quarterly Journal of Applied Mathematics II (2), 164-168, 1944
.

[Luntz & Brailovsky 69] A. Luntz & V. Brailovsky,
*On estimation of characters obtained in statistical procedure of recognition in russian*,
Technicheskaya Kibermetica, 3, 1969
.

[MacKay 92a] D. MacKay,
*A practical bayesian framework for backpropagation networks*,

Neural Computation, 4, 1992
.

[MacKay 92b] D. MacKay,
*Information-based objective functions for active data selection*,
Neural Computation, 4, 590-604, 1992
.

[Marquardt 63] D. Marquardt,
*An algorithm for least squares estimation of nonlinear parameters*,
Journal of Soc. Indust. Appl. Math, 11, 2, 431-441, 1963
.

[Martinez 04] J.-M. Martinez,
*Réseaux de neurones - Méthodologie et applications 2e Edition*,
Eyrolles, ch3, 2004
.

[Melville & Mooney 04] P. Melville & R. Mooney,
*Diverse ensembles for Active Learning*,
Proc. of 21st International Conference on Machine Learning, Banf, Canada, 2004
.

[Miller 90] A. Miller,
*Subset Selection in Regression*,
Chapman and Hall, 1990
.

[Miller 64] R. Miller,
*A Trustworthy Jacknife*,
Annals of Mathematical Statistics, 35, 1594-1605, 1964
.

[Mitchell 74] T. Mitchell,
*An algorithm for the construction of D-optimal experimental designs*,
Technometrics, 16, 203-210, 1974
.

[Monari 99] G. Monari,
*Sélection de modèles non-linéaires par leave-one-out ; étude théorique et application des réseaux de neurones au procédé de soudage par points*, Thèse de doctorat,
Université Pierre et Marie Curie - Paris, 1999
.

[Monari & Dreyfus 00] G. Monari & G. Dreyfus,
*Withdrawing an example from the training set : an analytic estimation of its effect on a non-linear parametrised model*,
Neurocomputing, 2000
.

[Monari & Dreyfus 02] G. Monari & G. Dreyfus,
*Local overfiting control via leverages*, vol. 14,
Neural Computation, 2002
.

[Moody 94] J. Moody,
*Prediction Risk and Architecture Selection for Neural Networks*,
From Statistics to Neural Networks : Theory and Pattern Recognition Applications, NATO ASI Series F, Springer-Verlag, 1994
.

[Poggio *et al.* 85] T. Poggio, V. Torre & C. Koch,
*Computational vision and regularization theory*,
Nature, 317, 314-319, 1985
.

[Rao & Toutenburg 95] C. Rao & H. Toutenburg,
*Linear Models - Least squares and alternatives*,
Springer Series in Statistics, 1995
.

[Raviart & Thomas 83] P. Raviart & P. Thomas,
*Introduction à l'analyse numérique des équations aux dérivées partielles*,
Editions Masson, 1983
.

[Rumelhart & McClelland 86] D. Rumelhart & J. McClelland,
*Parallel Distributed Processing*,
MIT Press, Cambridge, MA, 1986
.

[Saltelli & Homma 92] A. Saltelli & T. Homma,
*Sensivity Analysis for Model Output, Performances of Black Box Techniques on Three International Benchmark Exercises*,
Computational Statistics & Data Analysis, 13, 73-94, 1992
.

[Saporta 90] G. Saporta,
*Probabilités.Analyse des données et statistique*,
Editions Technip, 1990
.

[Schohn & Cohn 00] G. Schohn & D. Cohn,
*Less is More : Active Learning with Support Vector Machines*,
Proceedings of 17th International Conference on Machine Learning, 839-846, San Francisco, CA, Morgan Kaufmann, 2000
.

[Seung *et al.* 92] H. Seung, M. Opper & H. Sompolinsky,
*Query by Committee*,
Proceedings of the fifth annual ACM workshop on Computational Learning Theory, 287-294, 1992
.

[Shapire 99] R. Shapire, «
A short introduction to Boosting », 1999
.

[Sung & Niyogi 92] K. Sung & P. Niyogi,
*Active Learning for Function Approximation*,

Advances in Neural Information Processing Systems, 4, Morgan Kaufmann, 1992
.

[Thrun & Möller 92] S. Thrun & K. Möller,
*Active Exploration in Dynamic Environnements*,
Advances in Neural Information Processing Systems, 4, Morgan Kaufmann, 1992
.

[Tikhonov & Arsenin 77] A. Tikhonov & V. Arsenin,
*Solutions of Ill-Posed Problems*,
Winston, 1977
.

[Vapnik 98] V. Vapnik,
*Statistical Learning Theory*,
John Wiley & Sons, 1998
.

[Vapnik & Chervonenkis 71] V. Vapnik & A. Chervonenkis,
*On the uniform convergence of relative frequencies of events to their probabilities*,
Theory of Probability and its Applications, 1971
.

[Vila 85] J. Vila,
*Etude et comparaison de critères de plans d'expériences optimaux pour l'estimation des paramètres d'un modèle de régression non linéaire*, Thèse de doctorat,
Université Paris XI-Orsay, 1985
.

[Vila 86] J. Vila,
*Optimal designs for parameters estimation in nonlinear regression models : exact criteria*,
Invited Lecture XIIIth IBC, Seattle, 1986
.

[Vila 91] J. Vila,
*Local optimality of replications from a minimal D-optimal design in regression : a sufficient and a quasi-necessary condition*,
J. of Stat. Planning and Inference, 29, 261-277, 1991
.

[Vila & Gauchi 07] J. Vila & J. Gauchi,
*Optimal designs based on exact confidence regions for parameter estimation of a nonlinear regression model*,
Journal of Statistical Planning and Inference, 137, 9, 2935-2953, 2007
.

[Werbos 74] P. Werbos,
*Beyond regression : new tools for prediction and analysis in the behavioural sciences*, Thèse de doctorat,
Harvard University, 1974
.

[Witczak 06] M. Witczak,
*Toward the training of Feed-Forward Neural Networks with the D-optimum Input Sequence*,
IEEE Transaction on Neural Networks, 17, 2, 2006
.

[Wonnacott & Wonnacott 95] T. Wonnacott & R. Wonnacott,
*Statistique*,
Economica, 1995
.

[Wynn 70] H. Wynn,
*The sequential Generation of D-optimum Experimental Designs*,
Ann of Math Statisti, 41, 1655-1664, 1970
.