Université de Montréal

Les algorithmes d'apprentissage appliqués aux risques financiers

par
Charles Dugas

Département d'informatique et de recherche opérationnelle
Faculté des Arts et des Sciences

Thèse présentée à la Faculté des études supérieures
en vue de l'obtention du grade de Philosophiae Doctor
en Informatique

Juillet 2003

Université de Montréal
Faculté des études supérieures

Cette thèse intitulée
Les algorithmes d'apprentissage appliqués aux risques financiers

présentée par
Charles Dugas

a été évaluée par un jury composé des personnes suivantes :

Balázs Kégl

---

président-rapporteur

Yoshua Bengio

---

directeur de recherche

Douglas Eck

---

membre du jury

Hugh Chipman

---

examinateur externe

Silvia Gonçalves

---

représentante du doyen de la FÉS

Table des matières

Liste des tableaux

Table des figures

Liste des abréviations

$E[X]$	espérance mathématique de X.
$\mathbf{R}$	ensemble des nombres réels.
$\mathbf{R}^+$	ensemble des nombres réels supérieurs ou égaux à zéro
$\mathcal{U}(a,b)$	Distribution uniforme sur l'intervalle $[a,b]$.
$\mathcal{N}(\mu,\sigma)$	Distribution gaussienne centrée en μ et de variance σ^2.
$\vec{x}$	Un vecteur.

Remerciements

Pour leur soutien financier, je remercie les fonds FCAR et CRSNG, le CIRANO, la Society of Actuaries et l'Université de Montréal. Pour les conseils scientifiques, je remercie mon directeur de recherche Yoshua Bengio et mes collègues (devenus partenaires) Nicolas Chapados et Pascal Vincent. Merci aux membres du support informatique du DIRO pour leur travail efficace. Je remercie mes amis qui ont été près de moi pendant ces quelques années, en particulier Sébastien Sigouin, Christian Fournier, Hugo Fortin, Pierre St-Georges, Alexandre Viau pour leur support et leur intérêt. Je désire remercier mes parents Charles et Suzanne et ma soeur Rachel qui m'ont toujours encouragé.

Abstract

This thesis begins with an introductory chapter to the subject of machine learning. Then, three important contributions to the field of financial predictions, brought forth by the author and his coauthors, are presented. In the first paper, we define a model selection criterion, similar to cross validation, but that could be applied to sequential data with an underlying process that is potentially dynamic and poised with nonstationarities. We use a technique of continuous model selection criterion optimization with respect to hyper-parameters in order to weight historical data used to estimate model parameters for the prediction of first and second moments of canadians stock returns. In the second paper, we define new classes of functions, similar to neural networks but that incorporate certains properties, known *a priori*, on the derivatives of the function we are attempting to approximate. We have developed proof of universality of the proposed classes of functions. With artificial data, we have analyzed the gains obtained in terms of bias and variance, by the use of the proposed classes. Also, we have obtained improvements on S&P500 call option pricing. Finally, the third publication is concerned with the use of neural networks for automobile insurance ratemaking and the optimization of risk cessions within the context of risk sharing pools.

Keywords: model selection, hyper-parameters, volatility forecasting, neural networks, call options, universal approximator, automobile insurance, risk sharing pools, mixture of experts.

Chapitre 1

Introduction à l'apprentissage statistique

Le but de ce chapitre est de mettre en relief les particularités des concepts et méthodes de l'apprentissage statistique qui distinguent cette approche des statistiques classiques. Par le biais d'une revue historique, la section 1.1 mène à la description des concepts fondamentaux de l'apprentissage statistique. La section 1.2 présente une taxonomie de l'apprentissage statistique et permet de situer la régression dans ce contexte. La problématique de la régression est formellement définie à la section 1.3 et se termine par un exemple (un test d'hypothèse) où les deux approches (classique et apprentissage) se confrontent dans leurs fondements. La notion de capacité d'un ensemble de fonctions et le dilemme biais-variance sont présentés aux sections 1.4 et 1.5, respectivement. La méthode de régularisation fait l'objet de la section 1.6. Les méthodes de sélection de modèles par validation croisée (section 1.7) et calcul de bornes (section 1.8) sont ensuite abordées. Finalement, nous concluons ce chapitre à la section 1.9.

1.1 Concepts

L'inférence statistique se concentre sur le problème suivant : étant donné une série de données empiriques provenant d'une dépendance fonctionnelle inconnue quelconque, il faut répondre à certaines questions liées à la connais-

sance de cette dépendance.[1]

Fisher (34; 35; 36; 37; 2) a développé le cadre des statistiques paramétriques et suggéré une méthode pour l'approximation des valeurs inconnues des paramètres d'un modèle : la méthode du maximum de vraisemblance. Glivenko (42), Cantelli (17) et Kolmogorov (63) ont utilisé une approche plus générale pour prouver que la distribution empirique converge de façon exponentielle vers la vraie distribution, quelle que soit cette vraie distribution. Ces deux développements fondamentaux peuvent être considérés comme les bases de deux approches philosophiquement divergentes de l'inférence statistique.

Le but de la première approche est d'*identifier* le processus générateur de données. À cette fin, on doit disposer d'une connaissance suffisante des lois physiques qui gouvernent le processus afin de construire un modèle qui y correspond. L'essence de cette approche de l'inférence statistique se résume donc à estimer les paramètres d'un modèle (présumément) connu en utilisant les données empiriques et par la suite, de développer des tests statistiques qui mènent au rejet ou non du modèle choisi (ou de certains de ses paramètres). Pour l'estimation des paramètres, on utilise souvent la méthode du maximum de vraisemblance qui jouit de propriétés asymptotiques avantageuses.

Selon la seconde approche, on tente simplement de *prédire* certaines propriétés des données futures en se basant sur les données disponibles. L'hypothèse sous-jacente est que la réalité est trop complexe pour être identifiée et capturée par un seul modèle. En particulier, les processus multivariés sont frappés de ce que Bellman (6) a baptisé la « malédiction de la dimensionalité » puisque le nombre de combinaisons de valeurs des variables croît de façon exponentielle avec la dimensionalité (le nombre de variables explicatives) du problème. Pour certains problèmes pratiques tels que l'assurance automobile ou la détection de fraudes sur les cartes de crédit, on considère des douzaines de variables explicatives. Dans ce contexte, l'hypothèse selon laquelle on peut identifier le processus générateur paraît à tout le moins naïve. La seconde approche est moins ambitieuse : étant donné un ensemble de données empiriques, et un ensemble de fonctions, on cherche une fonction qui approxime le comportement du processus générateur ou une fonction qui

[1] Dans le domaine de l'apprentissage statistique, on cherche souvent à estimer la dépendance fonctionnelle elle-même. Toutefois, il arrive que l'objectif visé soit d'obtenir un résultat *impliqué* par cette relation de dépendance. Or, le résultat cherché peut parfois être obtenu par une simple connaissance partielle de la relation de dépendance, d'où l'importance de la précision. Par exemple, en classification supervisée, on cherche $f(x) = \arg\max_y P(Y = y|x)$.

permet de répondre à une question dont la réponse dépend de ce processus. Le but est d'obtenir la meilleure performance possible sur des tâches prédictives, en utilisant de nouvelles données issues du même processus générateur. Cette seconde approche devra attendre l'avènement de l'ordinateur électronique, dans les années soixante, avant de connaître son essor.

Dans les années 60, Huber (51; 52; 53; 54; 55; 57) développe l'approche *robuste* pour les statistiques paramétriques. Dans les années 70, afin d'élargir l'ensemble des fonctions considérées, Nelder et Weddernburn (71) proposent les modèles linéaires généralisés qui ont récemment gagné en popularité dans la communauté actuarielle. Ces ensembles de fonctions élargis mènent au problème de la sélection de modèle (1). À partir des années 80, plusieurs chercheurs (14; 56; 75) commencent à considérer certains types de fonctions, nonlinéaires en leurs paramètres et avec moins d'hypothèses distributionnelles, en particulier les arbres de décisions et les réseaux de neurones. C'est aussi au cours de cette période que se développe une alternative à la méthode du maximum de vraisemblance : la méthode de régularisation (83), cette dernière étant adaptée au but opérationnel de l'apprentissage statistique.

Une branche de l'apprentissage statistique se concentre principalement sur le développement de raffinements des méthodes de régularisation (section 1.6) et de sélection de modèle (sections 1.7 et 1.8) afin d'améliorer les performances prédictives de certains algorithmes. Cette performance est souvent appellée *généralisation* : les algorithmes peuvent *apprendre* certaines relations entre les variables à partir des données fournies (*ensembles d'entraînement* et de *validation*) pour le choix d'un modèle et de ses paramètres. Ces relations s'appliquent à de nouvelles données (*ensemble de test* ou données hors-échantillon), i.e., elles sont générales et caractéristiques du processus générateur et non pas simplement particulières à un ensemble de données spécifique. Dans le cas de l'assurance automobile où les données ne peuvent être supposées i.i.d.[2] mais exhibent plutôt une structure séquentielle avec de potentielles non-stationarités (évolution du processus générateur dans le temps), la généralisation est obtenue par l'utilisation d'une méthodologie particulière : la validation séquentielle qui sera étudiée au chapitre 3.

[2]indépendantes et identiquement distribuées

1.1.1 Exemple : test d'hypothèse

Nous pouvons mettre en relief la divergence des deux approches à l'aide d'un exemple simple où ces dernières mènent à deux tests statistiques différents et donc possiblement à des conclusions différentes. Considérons le test statistique qui sert à rejeter ou non un paramètre d'une régression linéaire simple. Soit la relation $y_i = \alpha + \beta x_i + \xi_i$ où ξ_i est un bruit blanc, i.e., $E(\xi_i) = 0$, $Var(\xi_i) = \sigma^2$, $Cov(\xi_i, \xi_j) = 0$, si $i \neq j$. Le test classique pour rejeter la présence d'une influence de la variable indépendante X sur la variable observée Y est basé sur l'hypothèse nulle $H_0 : \beta = 0$, i.e., la valeur de *beta* est-elle vraiment égale à zéro? C'est la question posée par l'approche statistique standard. Selon l'approche préconisée en apprentissage statistique, la question serait plutôt : est-ce que le choix $\hat{\beta} = 0$ mènera à une meilleure généralisation que le choix $\hat{\beta} \neq 0$, calculé à partir de l'ensemble d'entraînement. Si nous définissons l'erreur de généralisation comme l'espérance de l'erreur quadratique (ESE) :

$$ESE = E[(Y - (\hat{\alpha} + \hat{\beta}X))^2]. \tag{1.1}$$

et que l'on s'intéresse à minimiser cette valeur, alors, tel que l'ont montré (41), le test statistique classique s'avère inapproprié et l'on doit plutôt se tourner vers un test hors-échantillon.

(70) illustrent quelques exemples de tests hors-échantillon non-biaisés. En particulier, on peut montrer que si la valeur de β est telle que le ratio signal-bruit β^2/σ^2 est inférieur à une certaine valeur positive θ, alors le choix $\hat{\beta} = 0$ nous permettra d'obtenir une meilleure généralisation. Si l'ensemble d'entraînement est composé de n valeurs *i.i.d.*
$(X_1, X_2, \ldots, X_n)$ pour la variable indépendante, alors la valeur du seuil est

$$\theta = \frac{E\left[\frac{(X-\bar{X})^2}{\sum_i (X_i-\bar{X})^2}\right]}{E[(X-\bar{X})^2]},$$

où $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$, est la moyenne échantillonnale et X est un exemple hors-échantillon. Donc, le test hors-échantillon nous mène à choisir $\hat{\beta} = 0$, i.e., une valeur nulle pour le paramètre de régression lorsque le ratio signal-bruit est trop faible et ce, malgré le fait que $\beta \neq 0$ (i.e., la vraie valeur sous-jacente n'est pas nulle) et que le test classique pourrait rejeter l'hypothèse nulle ($H_0 : \beta = 0$).

Ce que les tests empiriques de (41) démontrent est que si, dans l'exemple précédent, on s'intéresse réellement à savoir si $\beta = 0$, alors on doit utiliser le test classique puisque sa puissance est plus grande que le test hors-échantillon de généralisation. D'un autre côté, si l'on s'intéresse à la performance prédictive hors échantillon, le test classique s'avère trop libéral, i.e., qu'il mène au choix de $\hat{\beta} \neq 0$ trop souvent, ce qui peut avoir des conséquences dangereuses pour certaines applications prédictives et dans ce contexte, on doit utiliser un test hors-échantillon pour estimer l'erreur de généralisation.

Finalement, bien que la différence entre les deux types de statistiques devient faible à mesure que n tend vers l'infini, il faut noter que dans le cas de certaines applications avec beaucoup de variables d'entrée, l'effet de « petit échantillon » n'est pas négligeable pour deux raisons.

1. Lorsque le nombre de variables discrètes est grand, et que nous voulons tenir compte des interactions, le nombre de combinaisons de valeurs possibles devient rapidement très grand et donc, il y a en fait peu de données qui permettent de déterminer si le paramètre associé à une combinaison particulière est utile ou non.
2. Lorsque la distribution des données est très asymétrique (comme dans le cas de l'assurance automobile), les réclamations très élevées mais rares peuvent avoir un effet important sur l'estimation des paramètres, ce qui augmente la divergence entre les conclusions obtenues à l'aide des statistiques de l'ensemble d'entraînement et celles des statistiques de l'ensemble de validation et de test.

Une autre raison pour laquelle une approche opérationnelle est préférable dans le cas d'applications pratiques est que les tests hors-échantillon ne requièrent aucune hypothèse sur la forme distributionnelle des données[3]. En d'autres termes, la conclusion obtenue suite à l'utilisation d'un test classique pourrait être invalidée s'il s'avère que les données n'ont pas été générées par la classe de distributions paramétriques choisie. Cette relaxation des hypothèses distributionnelles que permet le cadre de l'apprentissage statistique découle d'une définition différente de la problématique de l'inférence statistique, ce qui a été mentionné brièvement au début de ce chapitre et que nous abordons plus formellement à la section 1.3, dans le cadre de la régression.

[3]Dans la plupart des tests, on fait simplement l'hypothèse que les données sont *i.i.d.*. Même cette hypothèse peut être relaxée afin de travailler avec des données sequentielles (72; 27; 16; 18)

1.2 Taxonomie

1.2.1 Apprentissage supervisé

Dans le cadre de l'apprentissage supervisé, chaque exemple d'apprentissage est formé d'une paire entrée(s)-sortie(s). Les entrées sont les variables explicatives du modèle et les sorties sont les variables dépendantes du processus. À partir d'ici nous nous attarderons au cas où le processus ne génère qu'une seule variable dépendante. La tâche consiste à apprendre la dépendance fonctionnelle entre les variables explicatives et la variable dépendante de sorte que, suite à un entraînement par le biais d'un algorithme d'apprentissage, le modèle choisi puisse, à l'aide des valeurs des variables explicatives d'un nouvel exemple de test, prédire avec une certaine précision la valeur de la variable dépendante du processus.

Il est important de noter que, pour la plupart des applications pratiques, on ne peut retrouver dans l'ensemble d'entraînement un exemple pour lequel toutes les valeurs des variables explicatives sont identiques à celles de l'exemple de test présenté au modèle[4], d'où la nécessité de développer des algorithmes d'apprentissage qui permettent de bien généraliser à de nouveaux exemples. Dans beaucoup d'applications, la valeur observée pour la variable dépendante est bruitée et donc la relation de dépendance entre les variables explicatives et la variable dépendante est stochastique, par opposition à une relation déterministe où, pour une combinaison de valeurs des variables explicatives, une et une seule valeur de la variable dépendante peut être observée. Donc, dans le contexte d'une relation de dépendance stochastique, si l'on arrivait à identifier un exemple d'entraînement parfaitement identique à l'exemple de test, pourrait-on se rabattre sur la valeur observée de la variable dépendante de l'exemple d'entraînement pour inférer celle de l'exemple de test ? La réponse est non et c'est là le problème qui justifie le développement d'algorithmes robustes, c.-à-d. qui permettent d'extraire l'information tout en filtrant le bruit présent dans les exemples d'apprentissage.

Le paradigme de l'apprentissage supervisé se subdivise lui-même en deux types principaux de problèmes : classification et régression. Dans le cadre de la classification, on doit établir à quelle catégorie appartient un exemple de test. Par exemple, dans le cas de la reconnaissance de caractères manuscrits en français, anglais et quelques autres langues, il y a 26 ou 36 classes,

[4]La simple présence d'une variable continue parmi les variables explicatives rend théoriquement impossible ce pairage entre deux exemples.

dépendamment si les chiffres sont inclus ou non. Un lecteur optique lit un caractère et le représente sous forme numérique. Ensuite, étant donné l'information provenant de la lecture, le modèle de classification doit identifier quelle lettre ou quel chiffre se trouve sous le lecteur. Pour ce qui est de la régression, la variable dépendante est réelle et peut donc prendre une infinité de valeurs. Le but est que la valeur prédite pour un exemple de test soit la plus proche possible de la valeur observée. Pour établir la proximité de deux valeurs, on choisit une métrique, généralement l'erreur quadratique. Dans cette thèse, seuls des problèmes de régression sont abordés.

1.2.2 Apprentissage Non-supervisé

Dans le cadre de l'apprentissage non-supervisé, les exemples d'apprentissage n'ont pas d'étiquette, i.e., pas de valeur dépendante. Le but est de découvrir une structure qui permet de décrire de façon concise la distribution des exemples, i.e., de résoudre le problème d'estimation de densité. La possibilité de découvrir une structure représentative est intimement liée à la métrique qui est utilisée pour quantifier la proximité de deux exemples. Cette métrique devrait servir à incorporer les connaissances *a priori* sur le processus observé. Par la suite, un algorithme d'apprentissage peut être utilisé pour découvrir la structure sous-jacente. Ces algorithmes font généralement partie de la famille des algorithmes de groupement (*clustering*) supposent que chaque exemple d'apprentissage est issu d'un certain groupe d'exemples répartis autour d'un centroïde. Les algorithmes de groupement tentent de découvrir l'emplacement de chacun des centroïdes et la dispersion des exemples autour de ce centroïde. Plus la similarité entre les exemples d'un même groupe est grande par rapport à la similarité entre des exemples provenant de groupes différents, plus on considère que l'on a réussit à découvrir une structure représentative. Généralement, le nombre de centroïdes est fixé avant la phase d'optimisation et le nombre optimal de centroïdes est déterminé selon une méthode de sélection de modèle. Plusieurs tentatives d'automatiser le choix du nombre de centroïdes et de l'intégrer à la phase d'optimisation ont été réalisées. L'appartenance des exemples aux centroïdes est considérée comme étant une variable cachée du processus. Beaucoup d'algorithmes d'optimisation font partie de la classe des EM (« expectation-maximization ») (26). Notons aussi que l'estimation de densité et la réduction de la dimensionalité sont des problèmes importants de l'apprentissage non-supervisé.

1.2.3 Apprentissage par Renforcement

Dans le cadre de l'apprentissage par renforcement, l'utilisateur interagit avec le processus et les actions qu'il choisit de poser ont un effet sur le processus. Étant donné un contexte particulier du système, le but est de choisir une action parmi un ensemble d'actions possibles afin de maximiser un certain gain qui est reçu à la fin d'une séquence d'interactions. L'algorithme d'apprentissage doit apprendre à estimer, en rétrospective, la pertinence de chacune des actions qui ont été posées en regard d'un contexte particulier et ce, étant donné un gain qui n'a été reçu qu'à la fin de la séquence d'actions.

Dans le domaine financier, on peut placer le problème de la gestion active de portefeuille dans le contexte de l'apprentissage par renforcement : étant donné une série de transactions effectuées à différents moments, on tente de maximiser la valeur finale du portefeuille, au bout d'une certaine période. Étant donné la présence de frais de transactions, chercher à optimiser la valeur du portefeuille à chaque instant pourrait mener à un trop grand nombre de transactions et donc, au terme de la période, un résultat sous-optimal, une fois les frais déduits.

Un des plus grands défis de l'apprentissage par renforcement est de solutionner le dilemme exploration-exploitation. L'algorithme d'apprentissage peut exploiter les connaissances acquises en choisissant des actions qui ont été prises dans le passé et qui se sont avérées profitables puisqu'elles ont mené, en bout de ligne, à des gains importants. D'un autre côté, l'algorithme peut choisir d'explorer de nouvelles avenues en choisissant des actions qui n'ont jamais été prises pour obtenir de nouvelles connaissances. Il n'y a pas de solution générale à ce dilemme mais il est tout de même clair que la solution optimale à tout problème doit nécessairement combiner exploration et exploitation. L'apprentissage par renforcement n'est pas abordé dans cet ouvrage. Le livre de (81) propose une excellente introduction à ce sujet.

1.3 Problématique appliquée au cas de la régression

Dans cette section, nous développons, de façon plus formelle. la problématique de l'apprentissage statistique lorsqu'appliquées au contexte de la régression.

Soit un ensemble d'apprentissage ou ensemble d'entraînement $D_l = \{\vec{z}_1,$

$\vec{z}_2, \ldots, \vec{z}_l\}$ qui est un ensemble de données observées. Dans le cas de l'apprentissage supervisé, on a que chaque *exemple d'apprentissage* $\vec{z}_i$ est formé d'une paire $\{\langle \vec{x}_i, y_i \rangle\}_{i=1}^{\ell}$. Les éléments du vecteur $\vec{x}_i \in \mathbf{R}^d$ sont appelées variables explicatives et $y_i \in \mathbf{R}$ est la variable dépendante que l'on cherche à approximer.

Minimisation du Risque ou Erreur de généralisation

Le but de l'apprentissage statistique est de trouver une fonction f qui donne de bonnes prédictions de la valeur de y lorsqu'on lui présente un vecteur $\vec{x}$ quelconque. Pour mesurer la qualité des prédictions, on définit une métrique ou fonction de perte $L(y, f(\vec{x}))$. Pour les problèmes de régression, on utilise typiquement la fonction de perte quadratique $L(y, f(\vec{x})) = (y - f(\vec{x}))^2$. Pour mesurer la performance globale de la fonction f, on intègre la valeur de la fonction de perte sur l'espace des variables explicatives et de la variable dépendante pour obtenir la fonctionnelle de risque ou erreur de généralisation :

$$\begin{aligned} R(f) &= E_{X,Y}[L(y, f(x))] \\ &= \int_{X,Y} L(y, f(x))\, dF(x, y), \end{aligned} \tag{1.2}$$

où $F(x, y)$ est la fonction de distribution de probabilité définie sur le domaine $X \times Y$, évaluée au point $\langle x, y \rangle$. On suppose que les observations des variables d'entrée sont *i.i.d.*[5] selon une distribution inconnue $F(x)$ et que la relation entre la variable dépendante et les variables explicatives est stochastique et définie par la distribution inconnue $F(y|x)$. L'objectif est de minimiser cette erreur de généralisation. Malheureusement, la fonction de distribution de probabilité $F(x, y)$ est inconnue. On ne peut donc connaître la valeur exacte de $R(f)$ mais simplement l'approximer à l'aide des données disponibles.

Minimisation du Risque Empirique

L'ensemble d'entraînement D_l est tiré de la distribution inconnue $F(x, y)$. En utilisant cet ensemble, on peut approximer $R(f)$, telle que défini à l'équation 1.2, par le *risque empirique* :

$$R_{D_l}(f) = \frac{1}{l} \sum_{1}^{l} L(y_i, f(x_i)). \tag{1.3}$$

[5]Cette hypothèse sera relaxée au chapitre 3 alors que nous aborderons la modélisation de séries chronologiques.

En se basant sur le principe d'induction de la minimisation du risque empirique, on choisira la fonction $f^*_{D_l}$ qui minimise 1.3 :

$$f^*_{D_l} = \arg\min_f R_{D_l}(f). \tag{1.4}$$

Dans l'équation précédente, nous n'avons imposé aucune contrainte sur la classe de fonctions parmi laquelle nous pouvions choisir $f^*_{D_l}$. Or, si cette classe de fonctions est trop grande, on s'expose au risque de *surapprentissage*, c.-à-d. de choisir une fonction qui performe particulièrement bien pour les exemples de l'ensemble d'entraînement D_l mais ne *généralise* pas bien, c.-à-d. qui ne performe pas bien lorsque de nouveaux exemples lui sont présentés. Prenons le cas extrême où nous cherchons parmi toutes les fonctions de l'ensemble $\Omega = \{f : \mathbf{R}^d \rightarrow \mathbf{R}\}$. On pourra toujours utiliser la fonction

$$f^*_{D_l}(x) = \begin{cases} y_i & \text{si } x = x_i,\ x_i \in D_l, \\ 0 & \text{ailleurs.} \end{cases} \tag{1.5}$$

de sorte que $R_{D_l}(f^*_{D_l}) = 0$ puisque tous les exemples de l'ensemble d'entraînement ont été appris par coeur. Pour les valeurs de x autres que celles faisant partie de D_l, la fonction prédit une valeur arbitraire (0) et donc le processus d'apprentissage ne nous donne aucune information utile sur les autres points du domaine X. En fait, si nous n'imposons aucune restriction sur la classe de fonctions parmi laquelle sera choisie $f^*_{D_l}$, nous ne pourrons *apprendre* quoi que ce soit (77). Pour cette raison, nous limiterons notre recherche à une classe restreinte de fonctions $\mathcal{F}$:

$$f^*_{D_l} = \arg\min_{f \in \mathcal{F}} R_{D_l}(f) \tag{1.6}$$

Malgré cette restriction, le risque empirique reste un estimateur biaisé (optimiste) de l'erreur de généralisation, c.-à.-d. :

$$E_{D_l}[R_{D_l}(f^*_{D_l})] < E_{D_l}[R(f^*_{D_l})] \tag{1.7}$$

Afin d'obtenir un estimateur non biaisé mais bruité de l'erreur de généralisation de la fonction $f^*_{D_l}$, on pourra évaluer cette fonction sur un autre ensemble de données D'_l que l'on appellera *ensemble de validation* :

$$\hat{R}(f^*_{D_l}) = R_{D'_l}(f^*_{D_l}) \tag{1.8}$$

Enfin, considérons $f^* \in \mathcal{F}$, la fonction qui minimise l'erreur de généralisation :

$$f^* = \arg\min_{f \in \mathcal{F}} R(f). \tag{1.9}$$

Puisqu'on a généralement que $f^*_{D_l} \neq f^*$, alors

$$R(f^*) \leq R(f^*_{D_l}), \tag{1.10}$$

c.-à-d. que le choix $f^*_{D_l}$ est sous-optimal.

On souhaitera certainement trouver des méthodes qui permettent de choisir une fonction $f^*_{D_l}$ pour laquelle l'erreur de généralisation ne s'éloignera pas trop de la valeur minimale pouvant être obtenue. Autrement dit, on tentera de limiter l'écart $R(f^*_{D_l}) - R(f^*)$. Cette tâche revient à l'algorithme d'optimisation des paramètres.

Puisque ces valeurs ne peuvent être directement observées, on devra se contenter de les approximer. Ici, deux approches principales peuvent être envisagées. La première (section 1.7) consiste à approximer le risque par un estimateur non-biaisé mais bruité : le risque empirique mesuré sur un ensemble de validation.

Selon la seconde approche (section 1.8), on calcule des bornes probabilistes sur le risque étant donné la valeur du risque empirique tel que mesuré sur l'ensemble d'entraînement. À cette valeur du risque empirique, on ajoute un facteur de correction lié à la *capacité* classe $\mathcal{F}$ de fonctions. Dans le domaine de l'apprentissage statistique, la notion de capacité joue un rôle central. Elle sert à mesurer la complexité d'une classe de fonctions. Nous abordons cette notion à la section 1.4.

Ces approches permettent d'effectuer une sélection de modèle en espérant que ce choix s'approche du choix optimal basé sur les valeurs inconnues du risque. Notons que dans cet ouvrage, seule la première approche à l'estimation du risque est utilisée.

1.4 Capacité d'un ensemble de fonctions

La capacité d'un ensemble de fonctions est une mesure de l'étendue de cet ensemble dans l'espace des fonctions. Plus la capacité d'un ensemble est grande, plus il est possible d'y trouver des fonctions qui détectent les interactions utiles à la prédiction, ce qui est souhaitable. Toutefois, si cette capacité

est trop grande, il devient possible de trouver des fonctions qui modélisent les particularités d'un ensemble d'apprentissage spécifique, c.-à-d. le bruit. Il existe donc, pour chaque problème pratique, un niveau optimal de capacité. Le choix de ce niveau, à l'aide des outils fournis par l'apprentissage statistique, représente une part importante du travail d'un statisticien qui désire développer des modèles prédictifs performants. Plusieurs mesures de capacité existent, la mieux connue étant la dimension de Vapnik-Chervonenkis (84) ou VC dimension dont nous donnons la définition (77) pour le contexte de la classification.

Considérons un ensemble de m points. Si l'on considère un problème de classification binaire, alors il existe 2^m façon d'étiqueter ces m points.

Definition 1.4.1 *Soit une classe de fonctions $\mathcal{F}$. On dit que cette classe* **couvre** *un ensemble de m points si, pour chaque étiquetage de ces m points, il existe une fonction $f \in \mathcal{F}$ qui permet de les séparer correctement.*

Cette définition de couverture nous permet ensuite de définir la VC dimension :

Definition 1.4.2 *Soit m, le plus grand entier tel qu'il existe un ensemble de m points qui peut être couvert par $\mathcal{F}$, une classe de fonctions. Alors, on a que la* **VC dimension** *de $\mathcal{F}$ est égale à m, si m existe, ou ∞, sinon.*

Donc, une classe de fonctions $\mathcal{F}$ peut avoir une VC dimension infinie si pour toute valeur de m, il existe un ensemble de points qui peut être couvert par la classe $\mathcal{F}$. Un concept similaire de capacité infinie est celui d'*approximateur universel* :

Definition 1.4.3 *Une classe de fonctions $\hat{\mathcal{F}}$ de $\mathbf{R}^n$ vers $\mathbf{R}$ est un* **approximateur universel** *pour une classe de fonctions $\mathcal{F}$ de $\mathbf{R}^n$ vers $\mathbf{R}$ si, pour toute fonction $f \in \mathcal{F}$, tout domaine compact $D \subset \mathbf{R}^n$ et tout réel positif ϵ, on peut trouver $\hat{f} \in \hat{\mathcal{F}}$ telle que $\sup_{x \in D} |f(x) - \hat{f}(x)| \leq \epsilon$.*

Les réseaux de neurones à sortie linéaire et une couche cachée sont des approximateurs universels. Un des défis de l'apprentissage statistique est de développer des classes de fonctions qui permettent d'incorporer des connaissances *a priori* sur un processus tout en s'assurant que ces classes possèdent la propriété d'approximateur universel pour la classe de fonctions visée. Nous développons deux telles classes de fonctions au chapitre 6.

1.5 Dilemme Biais-Variance

Dans le cadre classique de la statistique, on cherche souvent à obtenir des estimateurs non-biaisés. Une fois cette propriété assurée, on cherche l'estimateur à variance minimale ou avec le taux de convergence le plus rapide. Dans les problèmes pratiques, le nombre d'exemples est limité. Pour cette raison, la propriété de convergence demeure un concept théorique et l'ajout d'un certain biais à un estimateur peut être bénéfique s'il se traduit par une réduction suffisante de la variance de l'estimateur. C'est là le *dilemme biais-variance* (39). Ce dilemme s'applique au choix de l'ensemble de fonctions $\mathcal{F}$: on pourra se limiter, de façon délibérée, à des ensembles ne contenant pas la fonction à approximer et ce, afin de pouvoir réduire la variance associée au choix de la fonction $f^*_{D_l}$. En réduisant la taille d'un ensemble de fonctions, on réduit sa capacité au sens de la section 1.4. Donc, le choix de la capacité optimale permet de résoudre le dilemme biais-variance.

Formellement, considérons un algorithme d'apprentissage qui, étant donné l'ensemble d'entraînement D_l, choisit la fonction $f^*_{D_l} \in \mathcal{F}$. Pour une réalisation particulière x de la variable aléatoire X, le biais de l'estimateur correspond à la différence entre l'espérance, sur tous les ensembles d'entraînement, de l'estimateur au point x et la vraie valeur du processus en ce point :

$$\text{biais}(x) \quad = \quad E_{D_l}[f^*_{D_l}(x)|X = x] - f(x). \tag{1.11}$$

Aussi, on voudra calculer la variance, par rapport à l'ensemble d'entraînement, de l'estimateur au point x :

$$\text{variance}(x) \quad = \quad E_{D_l}[(E_{D_l}[f^*_{D_l}(x)] - f^*_{D_l}(x))^2]. \tag{1.12}$$

Enfin, dans le cas de la perte quadratique, l'erreur de généralisation correspond à la somme du biais au carré et de la variance en plus d'un terme

de bruit :

$$
\begin{aligned}
\text{mse(x)} &= E_{Y,D_l}[(Y - f^*_{D_l}(x))^2] \\
&= E_{Y,D_l}[(Y - f(x) + f(x) - f^*_{D_l}(x))^2] \\
&= \underbrace{E_Y[(Y - f(x))^2]}_{\text{bruit}} + E_{D_l}[(f(x) - f^*_{D_l}(x))^2] \\
&\quad + 2\underbrace{E_{Y,D_l}[(Y - f(x))(f(x) - f^*_{D_l}(x))]}_{\text{zéro}} \\
&= \text{bruit}(x) + E_{D_l}[(f(x) - E_{D_l}[f^*_{D_l}(x)] + E_{D_l}[f^*_{D_l}(x)] - f^*_{D_l}(x))^2] \\
&= \text{bruit}(x) + \underbrace{(f(x) - E_{D_l}[f^*_{D_l}(x)])^2}_{\text{biais}^2} + \underbrace{E_{D_l}[(E_{D_l}[f^*_{D_l}(x)] - f^*_{D_l}(x))^2]}_{\text{variance}} \\
&\quad + 2\underbrace{E_{D_l}[(f(x) - E_{D_l}[f^*_{D_l}(x)])(E_{D_l}[f^*_{D_l}(x)] - f^*_{D_l}(x))]}_{\text{zéro}} \\
&= \text{bruit}(x) + \text{biais}^2(x) + \text{variance}(x). \qquad (1.13)
\end{aligned}
$$

En calculant l'espérance mathématique des valeurs précédentes (équations 1.11, 1.12, 1.13) par rapport à la variable aléatoire X, on obtient le biais, la variance et l'erreur quadratique moyenne de l'estimateur :

$$
\begin{aligned}
\text{biais} &= E_{X,D_l}[(f^*_{D_l}(x)] - f(x))^2], \qquad (1.14) \\
\text{variance} &= E_{X,D_l}[(E_{D_l}[f^*_{D_l}(x)] - f^*_{D_l}(x))^2], \qquad (1.15) \\
\text{mse} &= E_{X,Y,D_l}[(Y - f^*_{D_l}(x))^2]. \qquad (1.16)
\end{aligned}
$$

Le « bruit » provient du fait que l'on reconnaît le caractère stochastique (par opposition à déterministe) de la relation entre le processus observé et le vecteur X de variables explicatives. Un algorithme d'apprentissage ne peut aider à réduire le bruit, seulement le biais et la variance. Le seul moyen de réduire le bruit est de modifier le vecteur des variables d'entrées afin de permettre à l'algorithme d'utiliser une information plus riche. Certains processus sont fondamentalement déterministes : par exemple, en détection de fraude sur les cartes de crédit, on évalue la probabilité qu'une transaction soit frauduleuse *après* la transaction. Or, en théorie, si l'on pouvait, pour chaque transaction, prendre un échantillon d'ADN de l'acheteur et le comparer avec celui de la personne qui détient la carte, on disposerait d'une information à toutes fins pratiques parfaite pour prendre une décision. Au contraire, la prime pour une police d'assurance automobile doit être déterminée *avant* la

période de couverture et donc, quel que soit le niveau de raffinement des variables utilisées pour modéliser un conducteur, il restera toujours un aspect stochastique incontrôlable.

1.6 Régularisation

Le principe de régularisation (83) permet, au cours de la phase d'entraînement d'un modèle, d'introduire une préférence pour certaines fonctions de l'ensemble $\mathcal{F}$ en optimisant un critère d'apprentissage qui correspond généralement au risque empirique défini à l'équation 1.3 auquel on ajoute un *terme de régularisation* :

$$R_{D_l}(\lambda, f) = R_{D_l}(f) + \lambda\Omega(f) \tag{1.17}$$

où la fonctionnelle $\Omega(\cdot)$ établit la relation désirée de préférence parmi les fonctions de l'ensemble $\mathcal{F}$. Le coefficient λ, appelé *hyperparamètre*, sert à contrôler la force avec laquelle on impose cette préférence par rapport à la minimisation du risque empirique. La régularisation permet ainsi d'incorporer certaines connaissances connues *a priori* sur le type de fonction recherchée. La figure 1.1 illustre l'effet de la régularisation sur le processus d'optimisation.

Parmi les méthodes de régularisation les plus populaires, on trouve la régression *ridge* (48) qui consiste à pénaliser la norme L_2 des paramètres. Soit la fonction linéaire

$$f(\vec{x}) = \beta_0 + \sum_{i=1}^{p} \beta_i x_i, \tag{1.18}$$

le terme de pénalité sera

$$\Omega(f) = \sum_{i=1}^{p} \beta_i^2 \tag{1.19}$$

où β_i est le $i^{\text{ème}}$ paramètre de la régression. Lorsqu'appliquée aux réseaux de neurones cette méthode est connue sous le nom de *weight decay* :

$$\Omega(f) = \sum_{j=1}^{n_h} \sum_{i=1}^{p} w_{i,j}^2 \tag{1.20}$$

où n_h est le nombre d'unités cachées du réseau de neurones. Dans le cas de la régression *lasso*, on pénalise en tenant compte de la valeur absolue des paramètres. Cette contrainte est toutefois ferme :

$$R_{D_l}(\lambda, f) = R_{D_l}(f), \tag{1.21}$$

$$\text{sous la contrainte} \quad \sum_{i=1}^{p} |\beta_i| < \lambda. \tag{1.22}$$

Donc, selon la méthode *lasso* une fonction de l'ensemble $\mathcal{F}$ est soit admissible ou non, dépendamment si elle respecte ou non la contrainte stipulée. Parmi les fonctions admissibles, il n'existe aucun ordre de préférence. Selon la méthode de régularisation par *weight decay*, les fonctions sont ordonnées de façon continue entre elles.

Dans le cas de la régression, il est possible de donner une interprétation bayésienne à la méthode de régularisation qui correspond en fait à la supposition de l'existence d'une distribution *a priori* sur la distribution des paramètres. La régression ridge peut être associée à un *a priori* gaussien et la régression lasso à un *a priori* laplacien.

1.6.1 Exemple : Optimisation des paramètres

Considérons le même problème de régression tel que défini ci-haut mais cette fois, sous l'angle de l'estimation du paramètre de la régression. L'objectif est de minimiser l'erreur de généralisation espérée ce qui ne veut pas dire qu'il faille minimiser l'erreur de quadratique moyenne (MSE) sur l'ensemble d'entraînement :

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - (\hat{\alpha} + \hat{\beta}x_i))^2. \tag{1.23}$$

Pourtant, dans le cadre de l'approche classique et de l'utilisation de la méthode du maximum de vraisemblance, c'est bien la minimisation de la MSE telle qu'évaluée sur l'ensemble d'entraînement et définie à l'équation 1.23 qui est l'objectif visé.

Selon les principes de l'apprentissage statistique tels que définis plus haut on cherche à obtenir une meilleure généralistion, ce qui nous mène vers

une méthode d'optimisation différente : la méthode de régularisation. Selon cette méthode, on choisit de minimiser un critère pénalisé, i.e., on utilise la régression ridge :

$$MSE_\lambda = \frac{1}{N}\sum_{i=1}^{N}(y_i - (\hat{\alpha} + \hat{\beta}x_i))^2 + \lambda\hat{\beta}^2 \tag{1.24}$$

avec $\lambda \geq 0$ et le minimum est obtenu pour $\beta = \hat{\beta}_\lambda$. Ce minimum est toujours avec $||\hat{\beta}_\lambda|| < ||\hat{\beta}_0||$ où $\hat{\beta}_0$ est l'estimateur OLS. Cette solution est biaisée : $||E(\hat{\beta}_\lambda)|| < ||\beta||$ puisque $E[\hat{\beta}_0] = \beta$ si le modèle est juste.

Si le processus observé correspond réellement à une transformation affine des entrées avec bruit gaussien σ^2, alors on peut facilement montrer que la valeur optimale pour λ est

$$\lambda^* = \frac{\sigma^2}{N||\beta||^2}. \tag{1.25}$$

Donc, le modèle optimal (au sens de la minimisation de l'erreur de généralisation) est biaisé.

Une augmentation de la valeur de λ correspond à une réduction de la capacité d'un modèle au sens de la VC-dimension (84). Si la valeur choisie pour λ est trop élevée, il en résulte un sous-entraînement et un biais trop élevé. À l'inverse, une valeur trop faible de λ mène à une surentraînement et une variance trop élevée. Autrement dit, la valeur de λ sert à résoudre le dilemme biais-variance (39).

La solution de l'équation 1.25 n'est pas pratique puisqu'elle requière la connaissance de β, que nous cherchons à estimer. Aussi, elle ne s'applique que si les données sont distribuées selon une loi gaussienne. Le choix de la valeur de λ nous sera fournie par des algrithmes de sélection de modèle. Heureusement, la valeur espérée de l'erreur de généralisation possède un minimum global unique en fonction de λ. La formule ci-haut ne peut être utilisée aux fins de sélection de modèle et il faut se rabattre sur des méthode liées à la validation croisée que nous abordons à la prochaine section 1.7.

1.7 Validation Croisée

Au cours des dernières décennies, plusieurs méthodes servant à contrôler la capacité d'un modèle prédictif ont été développées. Elles font partie de ce

que l'on appelle les méthodes de sélection de modèle. Elles se regroupent en deux importantes catégories : validation croisée et calcul de bornes.

La validation croisée (79; 80; 3) est une méthode simple qui permet de mesurer directement la valeur espérée de l'erreur de généralisation. Le principe consiste à mettre de côté une portion de l'ensemble d'entraînement appelée ensemble test qui servira à mesurer la performance du modèle entraîné. On peut raffiner la procédure en subdivisant l'ensemble d'entraînement en K sous-ensembles. Ensuite, on choisit tour à tour chacun des sous-ensemble que l'on met de côté pour évaluer la performance hors-échantillon du modèle entraîné sur les autres sous-ensembles. On peut enfin calculer la moyenne des erreurs de prédiction pour obtenir un estimateur de l'erreur de généralisation.

Le choix de la valeur de K nous mène vers un autre dilemme biais/variance. Si K est trop faible, alors chacun des ensembles d'entraînement est trop semblable à l'ensemble global et l'estimateur de l'erreur de généralisation possède une plus grande variance par rapport à l'ensemble d'entraînement dont nous disposons. À l'inverse, si la valeur de K est élevée, alors chacun des sous-ensembles est beaucoup plus petit que l'ensemble d'entraînement total et l'estimateur est biaisé. Ce biais dépend de la nature de la courbe de l'erreur d'entraînement par rapport à la taille de l'énsemble d'entraînement. En pratique, on choisit généralement une valeur de K qui varie entre 5 et 10.

La validation croisée intervient dans le processus de sélection de modèle de la manière suivante : on entraîne des modèles de différentes classes de fonctions. On estime l'erreur de généralisation en mesurant la performance des modèles entraînés. On choisit ensuite le modèle qui offre la meilleure performance hors-échantillon. Les classes de fonctions sont parfois imbriquées, i.e. $\mathcal{F}_1 \subset \mathcal{F}_2 \subset, \ldots \subset \mathcal{F}_n$. Dans ce cas, la validation croisée sert à choisir le niveau optimal de complexité, ou capacité.

Lorsque les données possèdent une structure séquentielle, souvent temporelle, comme c'est le cas dans les applications financières, on entraîne un modèle avec les données passées pour l'implanter et l'appliquer aux données futures. Dans ces cas, par la force des choses, la valeur de K est fixée à 1. C'est le cas de toutes expériences décrites dans ce document. Par exemple, considérons le cas où l'on désire développer un modèle qui sera utilisé pour le calcul de primes d'assurance automobile en 2004. Supposons que l'on dispose des données de 1998 à 2002, c.-à-d. 5 ans de données historiques. On entraînera des modèles de différentes capacités avec les données de 1998 à 2000 inclusivement. Ensuite, les modèles entraînés seront confrontés aux données de 2001. Le modèle le plus performant sur les données de 2001 sera choisi ;

c'est là le processus de sélection de modèle qui nous permet de déterminer la capacité optimale devant être utilisée pour une tâche de modélisation particulière. Enfin, le modèle choisi sera testé sur les données de 2002 afin d'obtenir un estimateur non biaisé de l'erreur de généralisation. Une fois la capacité de modèle choisie, on entraînera un autre modèle avec la même capacité mais en utilisant les données de 2000, 2001 et 2002. C'est ce dernier modèle qui sera opérationnel en 2004.

1.8 Calcul de bornes

Selon le principe de la validation croisée, on obtient un estimé non-biaisé, à l'aide d'ensemble validation, de l'erreur de généralisation et c'est cette valeur qui dicte la sélection de modèle. La sélection de modèle par calcul de bornes requiert un seul ensemble d'entraînement qui sert à mesurer le risque empirique et une mesure de la capacité de l'ensemble de fonctions utilisé. On choisit tout d'abord un quantile qui réflète l'aversion au risque du statisticien dans le cadre du problème devant être résolu. Ensuite, on considère le risque empirique comme la moyenne d'une série d'observations i.i.d. du risque associé à la fonction choisie. La mesure de capacité nous permet de calculer la valeur du quantile de la distribution du risque de la fonction et c'est cette valeur qui sert de borne probabiliste sur le risque. Finalement, on utilise généralement une règle de type minimax pour la sélection de modèle.

Le cas le plus simple se produit lorsque l'ensemble $\mathcal{F}$ de fonctions considéré ne contient qu'un nombre fini N de fonctions. Alors, la capacité de $\mathcal{F}$ est égale à N. Nous développons ici les bornes pour ce cas simple lorsque appliqué dans un contexte de classification.

L'inégalité de Chernoff (20)[6] nous permet de caractériser, pour la classification, la convergence du risque empirique vers le risque :

$$P\{|R(f) - R_{D_l}(f)| \geq \epsilon\} \quad \leq \quad 2\exp(-2l\epsilon^2). \tag{1.27}$$

[6]Dans le cadre de la régression, on utilisera la généralisation de ce résultat fournie par Hoeffding (47) :

$$P\{R(f) - R_{D_l}(f) \geq \epsilon\} \quad \leq \quad \exp\left(-\frac{2l\epsilon^2}{(B-A)^2}\right). \tag{1.26}$$

où A et B sont des bornes sur la fonction de perte.

Et pour un ensemble de N fonctions,

$$\begin{aligned} & P\left\{\sup_{1\leq k\leq N}(R(f_k)-R_{D_l}(f_k))\geq\epsilon\right\} \\ & \leq\sum_{k=1}^{N}P\left\{R(f_k)-R_{D_l}(f_k)\geq\epsilon\right\} \\ & \leq N\exp(-2l\epsilon^2). \end{aligned} \tag{1.28}$$

Posons $\eta = N\exp(-2l\epsilon^2)$ de sorte que, avec probabilité $1-\eta$, on a que

$$R(f_k) \leq R_{D_l}(f_k)+\sqrt{\frac{\ln N-\ln\eta}{2l}} \tag{1.29}$$

simultanément, pour les N fonctions de l'ensemble $\mathcal{F}$ et en particulier pour la fonction choisie par le processus d'optimisation des paramètres pour les données de l'ensemble d'entraînement.

Dans le cas d'ensemble comprenant un nombre infini de fonctions, on doit remplacer N par une mesure de capacité telle la VC dimension définie plus haut (section 1.4) et les formules pour les bornes diffèrent un peu de celles obtenues dans cette section.

Le problème majeur de la sélection de modèle basée sur le calcul de bornes est d'ordre pratique. En particulier, notons que les bornes obtenues sur les probabilités d'erreur de classification sont régulièrement supérieures à un. Dans cet ouvrage, nous nous contentons d'utiliser le principe de validation croisée pour effectuer la sélection de modèle.

1.9 Conclusion

Ce chapitre devrait permettre au lecteur de situer cette thèse dans le cadre plus large des algorithmes d'apprentissage et de saisir les principes et les notions essentiels du domaine de l'apprentissage statistique. Les chapitres suivants décrivent diverses réalisations fructueuses de l'application de l'apprentissage statistique à l'evaluation de certains risques financiers.

Au chapitre 3, nous utilisons un principe d'optimisation continue des hyperparamètres pour améliorer la prédiction de la volatilité des actions sur les marchés canadiens. Au chapitre 6, nous développons deux classes de fonctions qui sont des approximateurs universels. En réduisant l'espace des fonctions à

ces classes plus restreintes, nous améliorons la valorisation d'options d'achat européennes sur l'indice S&P500. Au chapitre 9, une mixture d'experts est utilisée pour la tarification en assurance automobile. Ces trois applications fructueuses des algorithmes d'apprentissage à divers risques financiers nous permettent d'envisager un important potentiel commercial pour l'application des technologies développées au sein de la communauté de chercheurs de l'apprentissage statistique.

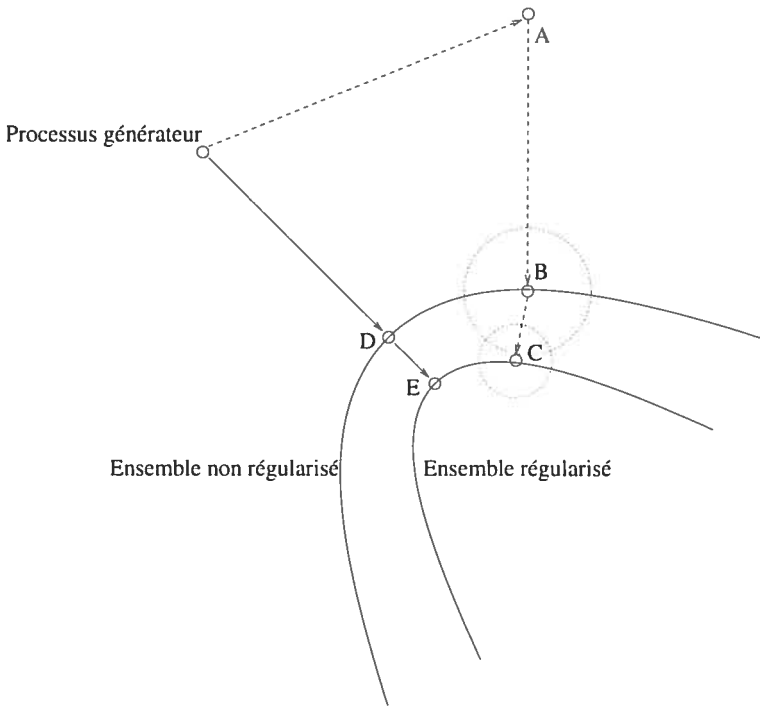

FIG. 1.1 – Illustration du principe de régularisation et du dilemme biais-variance. L'ensemble du graphique représente une projection en deux dimensions de l'espace des fonctions : $\Omega = \{\mathbf{R}^p \rightarrow \mathbf{R}\}$. Le processus générateur correspond à la fonction que l'on cherche à approximer. En utilisant le principe de minimisation du risque empirique sans contrainte sur l'espace de fonctions, par exemple, en choisissant la fonction décrite par l'équation 1.5, on obtient la fonction située au point A. Les fonctions des points B et C sont obtenues en contraignant l'optimisation aux ensembles non-régularisé et régularisé, respectivement. Les points D et E correspondent aux fonctions les plus rapprochées du processus générateur pour les ensembles non-régularisé et régularisé, respectivement. Les deux cercles concentriques pointillés servent d'échelle pour illustrer la différence de variance des deux ensembles. La régularisation permet une réduction de variance en contre-partie d'un biais accru. De façon générale, plus les données seront bruitées, plus le point A sera dispersé loin du processus générateur et plus il sera bénéfique d'augmenter la valeur de l'hyperparamètre de régularisation.

Première partie

Premier article

Chapitre 2

Introduction

Dans le cadre de l'optimisation des paramètres de modèles prédictifs, on minimise la valeur d'un critère d'entraînement qui s'exprime en fonction de l'erreur sur un exemple d'entraînement et de certains hyper-paramètres, qui restent fixes au cours de la minimisation. Par exemple, dans le cadre de la régularisation (83; 73), un hyper-paramètre contrôle la force du terme de pénalité et donc la capacité (84) du système. Lorsqu'un seul hyper-paramètre est utilisé, on peut aisément analyser son effet sur un critère de sélection de modèle (différent d'un critère d'entraînement et qui sert à choisir les valeurs des hyper-paramètres). Plusieurs critères pour le choix de la valeur des hyper-paramètres ont été proposés dans la littérature de la sélection de modèle, en général basés sur un estimateur de l'erreur de généralisation ou une borne probabiliste sur celle-ci (85; 1; 21). (7) a introduit une nouvelle approche pour l'optimisation simultanée de plusieurs hyper-paramètres, basée sur le calcul du gradient d'un critère de sélection de modèle par rapport aux hyper-paramètres. Dans l'article qui suit, nous appliquons cette approche à la modélisation des données de séries chronologiques non stationnaires, et présentons plusieurs expériences comparatives sur les rendements des actions canadiennes, des indices sectoriels et de l'indice TSE300.

Dans l'article, nous considérons des données séquentielles provenant d'un processus inconnu, de sorte que les observations ne sont pas nécessairement i.i.d.. La plupart des approches en inférence statistique et algorithmes d'apprentissage font l'hypothèse que les données sont i.i.d. (p.ex. (84)). Une exception est la séries de travaux de Warmuth et al. (p.ex. (65; 46)), dans lesquels la performance du modèle (qui est modifié à chaque instant) est comparée à celle d'un modèle de référence qui minimise la perte sur l'ensemble des

données jusqu'à la fin de la séquence. Dans l'article, nous ne considérons pas les bornes sur l'erreur de généralisation mais simplement des estimateurs de l'erreur de généralisation et des moyens de l'optimiser explicitement. En particulier, nous utilisons une extension de la validation croisée qui peut être appliquée aux données séquentielles non i.i.d. Toutefois, l'approche générale pourrait s'appliquer (peut-être avec plus de succès) à d'autres critères de sélection de modèle, puisque nous croyons que les estimateurs basés sur la validation croisée qui sont utilisés dans les expériences de l'article sont très bruités.

Dans la section 3.2, nous formalisons une notion d'erreur de généralisation pour des données séquentielles non i.i.d. qui est similaire à la notion proposée par (33). et décrivons une valeur analogue à la validation croisée pour l'estimation de cette erreur de généralisation que nous appellons validation séquentielle.

Dans la section 3.3, nous décrivons les résultats théoriques obtenus dans (7) pour le calcul du gradient d'un critère de sélection de modèle (p.ex. la validation séquentielle) par rapport aux hyper-paramètres. En particulier, nous considérons des hyper-paramètres qui contrôlent la longueur de la fenêtre de temps dont les données historiques serviront à l'entraînement d'un estimateur, au temps t, afin de prendre une décision concernant une variable qui sera observée au temps $t + 1$.

Dans la section 3.4, nous décrivons des expériences qui ont été réalisées sur des données artificielles qui ont été générées par un processus non stationnaire (avec un changement abrupte dans la distribution des données). Dans la section 3.5, nous décrivons les expériences réalisées sur des données financières : prédire les premier et second moments pour les actions canadiennes, les indices sectoriels et l'indice TSE300.

Cet ouvrage fut publié en tant que rapport technique (8). L'auteur fut le rédacteur principal de l'article et a contribué aux expériences réalisées.

Chapter 3

Learning Simple Non-Stationarities with Hyper-Parameters

test: We consider sequential data that is sampled from an unknown process, so that the data are not necessarily *i.i.d.* (independent and identically distributed). Most approaches to machine learning assume that data points are *i.i.d.*. Instead we consider a measure of generalization that does not make this assumption, and we consider in this context a recently proposed approach to optimizing hyper-parameters, based on the computation of the gradient of a model selection criterion with respect to hyper-parameters. Here we use hyper-parameters that control a function that gives different weights to different time steps in the historical data sequence. The approach is successfully applied to modeling the volatility of stock returns one month ahead. Comparative experiments with more traditional methods are presented.

3.1 Introduction

Many learning algorithms can be formulated as the minimization of a *training criterion* which involves both training errors on each training example and some *hyper-parameters*, which are kept fixed during this minimization. For example, in the regularization framework (83; 73), one hyper-parameter controls the strength of the penalty term and thus the capacity (86) of the system. When there is only a single hyper-parameter one can easily explore

how its value affects a *model selection criterion* (that is not the same as the training criterion, and is used to select hyper-parameters). Many criteria for choosing the value of the hyper-parameter have been proposed in the model selection literature, in general based on an estimate or a bound on generalization error (85; 1; 21). A new approach was introduced in (7) to simultaneously optimize many hyper-parameters, based on the computation Of the *gradient of a model selection criterion with respect to the hyper-parameters.* In this paper, we apply this approach to the modeling of non-stationary time-series data, and present several comparative experiments on financial returns data for Canadian stocks, sector indices, and the TSE300 market index.

In this paper we consider sequential data that is coming from an unknown process, so that it is not necessarily *i.i.d.*. Most approaches to machine learning and statistical inference from data assume that data points are *i.i.d.* (see e.g. (86)). One exception is the set of papers from Warmuth and others (e.g, (65; 46)), based on a comparison (and bounds) between the performance of the model (which is updated after each time step) and the performance of a reference model which minimizes the loss over all the data up to the end of the sequence. In this paper, we will not consider bounds on generalization error, but simply estimates of generalization error and ways to optimize it explicitly. In particular, we use an extension of the cross-validation criterion that can be applied to sequential non-*i.i.d.* data. However, the general approach could be applied (and maybe better results obtained) with other model selection criteria, since we suspect that the cross-validation based estimate that we used in our experiments is very noisy.

In Section 3.2, we formalize a notion of generalization error for data that are not *i.i.d.*, similar to the notion proposed in (33), and describe an analogue to cross-validation to estimate this generalization error, that we call **sequential validation**.

The contribution of this paper is described in Section 3.3, where we first summarize the theoretical results obtained in (7) for computing the gradient of a model selection criterion (such as for exemple the sequential validation estimate) with respect to hyper-parameters and then adapt this framework for the case where one deals with sequential data. We develop an algorithm (susection 3.3.2) for the simulatenous optimization of both parameters and hyper-parameters when dealing with sequential data. In particular, we consider hyper-parameters that smoothly controls how far in the past the historical data should be used for training a predictor at time t for making a decision concerning a variable which will be observed at time $t + 1$.

In Section 3.4, we describe experiments performed on artificial data that are generated by a non-stationary process (with an abrupt change in distribution). In Section 3.5, we describe experiments performed on financial data: predicting next month's return first and second moment, for Canadian stocks and stock aggregates.

3.2 Generalization Error for Non-IID Data

A *model selection criterion* is not the same as the training criterion: it is a criterion used to select hyper-parameters, more generally to compare and choose among models which may have a different capacity (86). Many model selection criteria have been proposed in the past (85; 1; 21). When there is only a single hyper-parameter one can easily explore how its value affects the model selection criterion: typically one tries a finite number of values of the hyper-parameters and picks the one with the lowest value of the model selection criterion. In this paper, we will use an estimate of generalization error for non-*i.i.d.* data that is similar to the cross-validation estimate often used in machine learning.

Let us consider a sequence of data points Z_1, Z_2, ..., with $Z_t \in \mathcal{O}$ (e.g., $\mathcal{O}$ = the reals $\mathcal{R}^n$) generated by an unknown process: $Z_t \sim P_t(Z)$. At each discrete time step t, in order to take a decision or make a prediction, we are allowed to choose a function f from a set of functions $\mathcal{F}$, and to do this we are allowed to use past observations $z_1^t = (z_1, z_2, \ldots z_t)$. Let us call this choice f_t. At the next time step (or more generally at some time step in the future), we will be able to evaluate the quality of our choice f_t, with a known cost function $Q(f_t, Z_{t+1})$.

For example, in this paper we consider $Z = (X, Y)$, an input/output pair with $X \in \mathcal{R}^n$ and $Y \in \mathcal{R}^m$ real-valued vectors. We will describe an algorithm for the case of affine functions (with a matrix of parameters θ), i.e., $\mathcal{F} = \{f : \mathcal{R}^n \rightarrow \mathcal{R}^m | f(\mathbf{x}) = \theta\tilde{\mathbf{x}}, \mathbf{x} \in \mathcal{R}^n, \theta \in \mathcal{R}^{m \times (n+1)}, \tilde{\mathbf{x}} = (\mathbf{x}, 1) = (x_1, \ldots, x_n, 1)\}$, and these functions will be selected to minimize the *squared error*

$$Q_s(f, (X, Y)) = 0.5(f(X) - Y)^2 \tag{3.1}$$

In this context, we can define the *expected generalization error* G_{t+1} at the next time step as the expectation over the unknown process P_{t+1} of the

loss function Q:

$$G_{t+1}(f) = \int_{z_{t+1}} Q(f, z_{t+1}) dP_{t+1}(z_{t+1}). \tag{3.2}$$

What we would like is to select f which has the lowest expected generalization error. How should we do this?

At time t, the *empirical error* of a choice f, when data z_1^t have been observed, is $\sum_{s=1}^{t} Q(f, z_s)$. According to an analogy with the principle of empirical risk minimization (86) for *i.i.d* data, we would choose the $f \in \mathcal{F}$ that minimizes the empirical error. However, if $\mathcal{F}$ has too much capacity (86), or if the sequence is "non-stationary", it might be better to choose an f that minimizes an alternative functional, e.g., a *training criterion* that penalizes the complexity of f and that may weight differently the different past observations:

$$C_t(f, z_1^t, \lambda) = R(f, \lambda) + \sum_{s=1}^{t} w_s(t, \lambda) Q(f, z_s) \tag{3.3}$$

where $\lambda \in \mathcal{R}^l$ is a vector of so-called **hyper-parameters**, $w_s(t, \lambda)$ is a scalar function of λ that puts a different weight on different training points, and $R(f, \lambda)$ is a penalty term that defines a preference over functions f within $\mathcal{F}$.

Let us suppose that one chooses f which minimizes the above C_t for a given choice of λ. We obtain

$$f = F(z_1^t, \lambda) = \arg\min_{f \in \mathcal{F}} C_t(f, z_1^t, \lambda) \tag{3.4}$$

where F maps the data z_1^t and λ into a function $f \in \mathcal{F}$.

3.2.1 Extension of Cross-Validation to Non-IID Data

One way to select the hyper-parameters λ in equations 3.3 and 3.4 is to consider *what would have been the generalization error* in the past if we had used the hyper-parameters λ. This is analogous to cross-validation, which selects the hyper-parameters for *i.i.d.* data by comparing the generalization error that would have been obtained on a *held-out subset* of data when the function f is chosen on a *training subset*. With cross-validation, a good (unbiased but noisy) estimate of generalization error can be obtained by averaging the error on held-out subsets, for a large number of training/held-out partitions (e.g., in 1-fold cross-validation all the possible training subsets

of size $t-1$ and test subsets of size 1 are considered). However, if the data points are not *i.i.d.*, we can't shuffle them: their order is important.

Therefore, we will use a **sequential cross-validation** criterion:

$$E_t(\lambda, z_1^t) = \frac{1}{t - M'} \sum_{s=M'}^{t-1} Q(F(z_1^s, \lambda), z_{s+1}) \tag{3.5}$$

where $f = F(z_1^s, \lambda)$ (eq. 3.4) minimizes the training criterion (eq. 3.3), and M' is the minimum number of training points to select a value of f within $\mathcal{F}$. **At each time** t, we will select the hyper-parameters that minimize E_t:

$$\lambda_t = \arg\min_{\lambda} E_t(\lambda, z_1^t) \tag{3.6}$$

This is similar to the principle of minimization of the empirical error, but applied to the selection of hyper-parameters, in the context of sequential data. Once λ_t has been selected, the corresponding f_t is therefore obtained:

$$f_t = F(z_1^t, \lambda_t) = \arg\min_{f} C_t(f, z_1^t, \lambda_t). \tag{3.7}$$

Note that the sequential cross-validation algorithm as defined in equation 3.5 runs in time $O(t^2)$. One could also set a limit M'' on the number of observations used so that the summation would run from $s = M'$ to $s = min(t, M'')$. In that case, the algorithm would run in time $O(t * M'')$.

3.3 Optimizing Hyper-Parameters for Non-IID Data

In this section we summarize the theoretical results already presented in (7) and extend the results obtained in the previous section for the application of interest here, i.e., using hyper-parameters for modeling possibly non-stationary time-series. We also present the algorithm for that purpose.

For parameterized classes of functions, each $f \in \mathcal{F}$ is associated with real-valued parameters θ. The minimization of the training criterion C_t (eq. 3.3) yields a value for the parameters θ. We will abuse notation and write θ instead of f in the expressions defined above (but there is a bijection between $\theta \in \mathcal{R}^q$ and $f \in \mathcal{F}$). If the functions of $\mathcal{F}$ are smooth in their parameters, the resulting θ depends on the choice of hyper-parameters in a continuous way:

$$\theta_t = \theta(z_1^t, \lambda) = \arg\min_{\theta} C_t(\theta, z_1^t, \lambda). \tag{3.8}$$

For the application considered in this paper, the hyper-parameters λ are used for controlling *pattern weights* on past data points, i.e., giving a weight $w_s(t, \lambda)$ to the past observation at time s when minimizing the training error up to time t, in equation 3.3. To weight the past data points, giving near 1 weight to recent time steps after a *threshold time step* λ_2, and near zero weight to earlier time steps (see also the curve of pattern weights in Figure 3.1):

$$w_s(t, \lambda) = \frac{1}{1 + \exp(-\lambda_1(s - \lambda_2))} \tag{3.9}$$

It is then straightforward to compute the derivatives $\frac{\partial w_s}{\partial \lambda}$, which will be useful in order to numerically optimize the hyper-parameters.

To apply gradient-based optimization to the selection of the hyper-parameters (eq. 3.6), we will compute the gradient with respect to the model selection criterion E_t (eq. 3.5):

$$\frac{\partial E_t}{\partial \lambda} = \sum_{s=M'}^{t-1} \sum_{i,j} \frac{\partial Q(\theta(z_1^s, \lambda), z_{s+1})}{\partial \theta_{i,j}(z_1^s, \lambda)} \frac{\partial \theta_{i,j}(z_1^s, \lambda)}{\partial \lambda}. \tag{3.10}$$

The derivative of the loss Q with respect to the parameters θ is generally easy to compute (since these gradients are used explicitly or implicitly to obtain θ). What is unusual here is that we need to compute the derivative of the parameters θ with respect to the hyper-parameters λ. Basically, this involves looking at the minimum of C_t with respect to the parameter vector θ, for a given hyper-parameter vector λ (equation 3.8), and seeing how a change in λ influences the solution θ.

In this paper, we will focus on the case in which Q is quadratic with respect to θ (e.g., with linear regression), but the formulae for the more general case can be found in (7). In both the quadratic case and the non-quadratic case, the computation of the model selection criterion with respect to the hyper-parameters requires the computation of the Hessian matrix of second derivatives of the training criterion with respect to the parameters:

$$H_s = \frac{\partial^2 C_s}{\partial \theta^2}.$$

In the quadratic case, to obtain $\theta_s = \theta(z_1^s, \lambda)$, one can minimize C_s by solving a linear system in θ,

$$-H_s\theta = \left.\frac{\partial C_s}{\partial \theta}\right|_{\theta=0} = b_s.$$

where the right-hand side b_s is the first term in the Taylor expansion of C_s with respect to θ. Since H_s is symmetric positive-definite, this can be done using a Cholesky decomposition of H_s:

$$H_s = L_s L_s'$$

where L_t is lower-diagonal (with zeros above the diagonal). Then one uses back-substitution twice to solve, first for u_s

$$L_s u_s = -b_s,$$

then for θ_s:

$$L_s' \theta_s = u_s.$$

The procedure described in detail in (7) shows how to back-propagate gradients through these three steps in order to compute, from $\frac{\partial E_t}{\partial \theta_s}$, first $\frac{\partial E_t}{\partial u_s}$ and part of $\frac{\partial E_t}{\partial L_s}$, then $\frac{\partial E_t}{\partial b_s}$, and finally the complete value of $\frac{\partial E_t}{\partial L_s}$ and $\frac{\partial E_t}{\partial H_s}$. Depending on the parametrization of the model and the hyper-parameters, one can then easily compute $\frac{\partial E_t}{\partial \lambda}$ from $\frac{\partial E_t}{\partial H_s}$ and $\frac{\partial E_t}{\partial b_s}$. In the case of an affine model with a quadratic criterion as in equation 3.3, we obtain

$$H_s = \frac{\partial^2 C_s}{\partial \theta^2} = \sum_{k=1}^{s} w_k(s,\lambda) \tilde{x_k} \tilde{x_k}'$$

where x_k is the k-th input vector in the input sequence, and similarly

$$-b_s = -\left.\frac{\partial C_s}{\partial \theta}\right|_{\theta=0} = \sum_{k=1}^{s} w_k(s,\lambda) y_{k+1} \tilde{x_k}$$

so the following gradients are obtained:

$$\frac{\partial E_t}{\partial \lambda} = \sum_{s=M'}^{t-1} \sum_{k=1}^{s} \frac{\partial w_k(s,\lambda)}{\partial \lambda} \sum_{i,j} \frac{\partial E_t}{\partial H_s(i,j)} \tilde{x}_{k,i} \tilde{x}_{k,j} + \sum_i \frac{\partial E_t}{\partial b_s(i)} \tilde{x}_{k,i} y_{k+1}. \quad (3.11)$$

In the more general case (non-quadratic case), the algorithm proposed in (7) requires computing the inverse of the Hessian. Note that an outer-product approximation of the inverse Hessian can be efficiently re-computed at each time step (without re-doing an inversion) using the algorithm described in (44).

We have outlined how one can compute the gradient of a model selection criterion E_t for data z_1^t with respect to the hyper-parameters λ. By using this gradient within an iterative optimization procedure (e.g., we use conjugate gradients in our experiments), one can therefore "train" the hyper-parameters λ_t with respect to the data z_1^t.

3.3.1 Measuring Generalization Error of the Model with Trained Hyper-Parameters

To estimate the generalization error of the procedure outlined above to find hyper-parameters, we need a second loop over time, in which the hyper-parameters are retrained after each time step, yielding λ_t, and used to learn some parameters $\theta(z_1^t, \lambda_t)$, and the parameters are used to evaluate generalization error $Q(\theta(z_1^t, \lambda_t), z_{t+1})$ on the next time step. Our estimate is

$$\hat{G} = \frac{1}{T-M} \sum_{t=M}^{T-1} Q(\theta(z_1^t, \lambda_t), z_{t+1}).$$

Here we have used the symbol M for the "minimum number of training points" for estimating BOTH the hyper-parameters and the parameters. Note that within each estimation of λ_t in the above sum, there is a sum over time steps s from M' to $t-1$ (as in equations 3.5 and 3.11), where M' is the "minimum of training points" for estimating the parameters θ (with fixed hyper-parameters λ). Clearly we need $M' < M$, and the difference $M - M'$ must be large enough to allow estimating the hyper-parameters (since the first M' time steps are not used in forming the validation errors in the sum E_t).

3.3.2 An Algorithm for training Hyper-Parameters

In order to illustrate the concepts defined above, we state here an algorithm for the implementation of the optimization of hyper-parameters when dealing with sequential data. We consider here the special case when using quadratic loss function and pattern-weight hyper-parameters. In the next sections, we report results on artificial data as well as financial data, using this algorithm to simultaneously optimize parameters and hyper-parameters.

```
initialize θ, λ
initialize G ← 0
for t = M', ..., T − 1
        for s = M ..., t
                compute ∂C/∂θ
                update θ
                set H, dHdl, b, dbdl ← 0
                for u = 0 ..., s
                        compute w_u, ∂w_u/∂λ
                        add w_u x_u x'_u to H
                        add (∂w_u/∂λ) x_u x'_u to dHdl
                        add w_u y_u x_u to b
                        add (∂w_u/∂λ) x'_u to dbdl
                compute H^{-1}, ∂H^{-1}/∂λ using H, dHdl
                compute ∂θ/∂λ using H, dHdl, b, dbdl
                compute ∂Q/∂λ
                compute ∂E/∂λ using ∂Q/∂λ and ∂θ/∂λ
                update λ
        add Q(z_{t+1}, θ)/(T − M' − 1) to G
return generalization error estimate G
```

3.4 Experiments on Artificial Data

To verify that the algorithm worked properly, we have tested it on artificially generated non-*i.i.d.* data with a single abrupt change in the input/output dependency at some point in the sequence. The single input / single output data sequence is generated by a Gaussian mixture for the inputs, and for the output given the input a left-to-right Input/Output Hidden Markov Model (9) with 4 states and two input/output distributions. These conditional distributions are conditional linear Gaussians (whose output expectation is a linear function of the input, and whose variance is fixed to 1). The first three states share the same input/output distribution and the last state has a different input/output distribution. When the model randomly switches from the 3rd state to the 4th state, the relation between the input and the output changes drastically, as seen on the left of Figure 3.1 (with the ratio of output to input). A sequence with 200 input/output pairs was

generated.

We compare a regular linear regression with a linear regression with three hyper-parameters: one hyper-parameter for the weight decay and two hyper-parameters (λ_1 and λ_2) to yield pattern weights on past data points with a sigmoidal decay (eq. 3.9). It means that training points before time λ_2 get little weight (little influence in the training criterion and the solution) while more recent training points do get most of the weight: λ_2 can be interpreted as the transition point of a change in regime. The hyper-parameter λ_1 controls how abrupt or smooth that transition is (see(eq. 3.9). In the extreme, when λ_1 is close to 0, all past training data points get an equal weight. At each time step after $t = 50$, the hyper-parameters are automatically selected by conjugate gradient descent to minimize the sequential cross-validation criterion (eq. 3.5), with gradients computed as explained in the previous section. The models are allowed to use the first 50 points for selecting their parameters (and hyper-parameters) for the first out-of-sample prediction, i.e., $M = 50$. Henceforth at each time step, (1) their prediction for the next time step is recorded, (2) the squared loss with respect to the corresponding output is recorded, (3) this input/output pair is added to their training data, and (4) their parameters (and hyper-parameters) are updated based on that enlarged training data. In the case of a model with hyper-parameters, the inner training uses at least $M' = 25$ points.

The out-of-sample MSE (Mean Squared Error) is the average of the squared loss on the predictions from $t = 51$ to the end of the sequence, $t = 200$. Each of these predictions at time t is truly out-of-sample since it is formed by only using data prior to t, both for selecting the parameters and the hyper-parameters. The out-of-sample MSE for the regular linear regression is 3.97 whereas it is only 0.62 when the hyper-parameters are optimized. As shown in Figure 3.1, much better performance is obtained with the adaptive hyper-parameters, which allow to quickly recover from the change in distribution at $t = 80$.

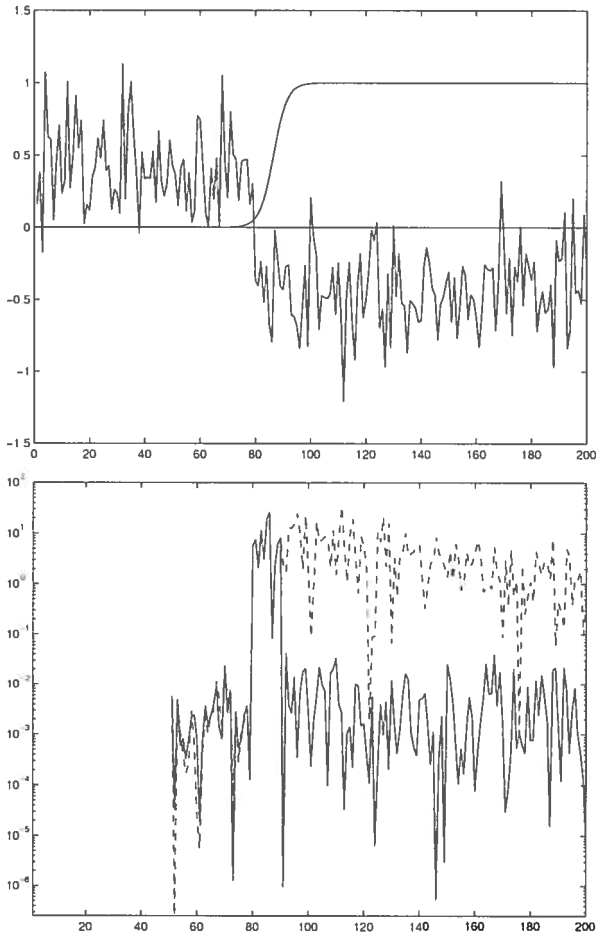

Figure 3.1: Top: ratio $\frac{y_t}{x_t}$ vs t from the generated data (noisy curve), and pattern weights $w_s(200, \lambda)$ (smooth curve) found at the end of the sequence by optimizing the hyper-parameters λ_1 and λ_2. Bottom: out-of-sample squared loss for each time step t, for the ordinary regression (dashed) and regression with optimized hyper-parameters (continuous).

3.5 Experiments on Financial Time-Series

In this section, we describe experiments performed on financial data: predicting next month's return first and second moment, for Canadian stocks and stock aggregrates. Let us introduce some notation. Let

$$r_t = value_t / value_{t-1} - 1$$

be a discrete return series (the ratio of the value of an asset at time t over its value at time $t-1$, which in the case of stocks includes dividends and capital gains). *In these experiments, our goal is to make predictions on the first and second moment of r_{t+1}, using information available at time t.*

These predictions could be used in financial decision taking in various ways: for asset allocation (taking risks into account), for estimating risks, and for pricing derivatives (such as options, whose price depends on the second moment of the returns).

In the experiments we directly train our models to predict these two moments, i.e., we are trying to learn

$$E_t[r_{t+1}]$$

and

$$E_t[r_{t+1}^2]$$

where E_t denotes the conditional expectation using information available up to time t. One way to learn these conditional expectations is to train the models to minimize the squared error (equation 3.1), with the output variable Y being next month's return or squared return, and X being an input variable that is chosen to represent a summary of currently available information (in addition, the parameters of the model also contain information about the past data).

Note that a good predictor for the first and second moment can be combined to obtain a predictor for the "volatility", e.g.,

$$E_t[(r_{t+1} - E_t[r_{t+1}])^2] = E_t[r_{t+1}^2] - E_t[r_{t+1}]^2 \tag{3.12}$$

We have performed experiments on three types of return series:

1. **Individual stocks**: monthly returns and monthly squared returns. We used data for 473 stocks from the Toronto Stock Exchange (TSE) for which at least 98 months of data were available, the earliest starting in January 1976, and the lastest data ending in December 1996.

2. **Sector returns**: monthly returns and monthly squared returns. We used data for the main 14 sectors of the TSE, from January 1956 to June 1997 inclusively.

3. **Market index returns**: monthly returns and monthly squared returns. We used data from the TSE300 stock index, from February 1956 to January 1996.

3.5.1 Experimental Setup and Performance Measures

In all the experiments, we compare several models on the same data, and we use the *sequential cross-validation* criterion (equation 3.5) to estimate generalization performance. The "minimum number of training points" for training both parameters and hyper-parameters is $M = 72$ months (6 years) in all the experiments. Following equation 3.5, after each time step, each model is re-trained (parameters and hyper-parameters) using the additional data point, and tested on the next month's returns observation. For the models with hyper-parameters, the "minimum number of training points" for training the parameters when the hyper-parameters are fixed is $M' = 48$ months (4 years).

In the experiments on stocks and sectors, there are several assets, with corresponding return series. Different models are trained separately on each of the assets, and the results reported below concern the average performance over all the assets. The chosen performance measure is the mean squared error (MSE): the average over out-of-sample predictions of half the squared difference between the prediction and the realized value. We have also estimated the variance of that average, and the variance of the difference between the MSE for one model and the MSE for a reference model, as described below. Using the latter, we have tested the null hypothesis that the two compared models have identical true generalization error.

For this purpose, we have used an estimate of variance that takes into account the dependencies among the errors at successive time steps. Let e_t be a series of errors (or error differences), which are maybe not *i.i.d.* Their average is

$$\bar{e} = \frac{1}{n} \sum_{t=1}^{n} e_t.$$

We are interested in estimating its variance

$$Var[\bar{e}] = \frac{1}{n^2} \sum_{t=1}^{n} \sum_{t'=1}^{n} Cov(e_t, e_{t'}).$$

Since we are dealing with a time-series, and because we do not know how to estimate independently and reliably all the above covariances, we will assume that the error series is covariance-stationary and that the covariance dies out as $|t - t'|$ increases. This can be verified empirically by drawing the autocorrelation function of the e_t series. The covariance stationarity implies that

$$Cov(e_t, e_{t'}) = \gamma_{|t-t'|}.$$

An unbiased and convergent estimator of the above variance is then the following:

$$\hat{v} = \frac{1}{n} \sum_{k=-\Delta_n}^{\Delta_n} (1 - |k|/\Delta_n)\gamma_k,$$

where $\lim_{n\to\infty} \Delta_n = \infty$ and $\lim_{n\to\infty} \Delta_n/n = 0$: we have used $\Delta_n = \sqrt{n}$.

The $\gamma's$ are estimated from the sample covariances (here for $k > 0$):

$$\gamma_k = \frac{1}{n-k} \sum_{t=1}^{n-k} e_t e_{t+k}.$$

Because there are generally strong dependencies between the errors of different models, we have found that much better estimates of variance were obtained by analyzing the differences of squared errors, rather than computing a variance separately for each average:

$$Var[\bar{e}^A - \bar{e}^B] = Var[\bar{e}^A] + Var[\bar{e}^B] - 2Cov[\bar{e}^A, \bar{e}^B]$$

where $\bar{e}^A$ is the average error on model A, etc...

In the tables below we give the p-value of the null hypothesis that model A is not better than model B (where model B is a reference model, usually the constant or Gaussian model). In the case that $\bar{e}^A < \bar{e}^B$, the alternative hypothesis is that A is better than B, so we use a single-sided test, with p-value $= \hat{P}(E[e^B] - E[e^A] \geq \bar{e}^B - \bar{e}^A)$. In the opposite case, we consider the converse test (with the alternative hypothesis being that B is better than A). To compute the p-values we assume that the average or the average difference is a normal variate, whose variance is estimated as explained above using the auto-covariances of the series.

3.5.2 Models Compared in the Experiments

The following models have been considered in the comparative experiments (note the "short name", in bold below, used in the tables):

- **Constant** model: this is the *reference* or *naive* model, which has *0 hyper-parameters* and only *1 free parameter* (which is the *historical average* of the past and current output observations). It is equivalent to an unconditional Gaussian model of the distribution of the observed variable (but we only care about the expectation, here). It is against this model that the other models are tested (p-values given in the tables).

- **Linear 1** model: this is a linear regression with 1 input, which is the current value of the output variable. There are *2 free parameters* and *0 hyper-parameters.*

- **Linear 2** model: this is similar to the **Linear 1** model, but there is an extra input which is the average over the last 6 months of the output variable. It has *3 free parameters* and *0 hyper-parameters.*

- **Linear 4** model: this is similar to the **Linear 2** model, but it also has as input the value of the output variable at the previous time step, as well as the average over the last 6 time steps of the return. This model was only used for the squared return prediction experiments. It has *5 free parameters* and *0 hyper-parameters.*

- **ARMA(p,q)** model: this is a recurrent model (Auto-Regressive Moving-Average) of orders p (number of auto-regressive recurrences) and q (number of moving average lags). In the case of predicting volatility (second moment), this is similar to a GARCH model except that there are no positivity constraints and we minimize only the prediction error on the squared return, rather than maximize the likelihood of the return. It has *1+p+q free parameters* and *0 hyper-parameters.* We have tried the following combinations of p and q: (1,1),(2,1),(1,2),(2,2).

- **Hyper Constant** model: this is like the **Constant** model except that there are weights on the past data (learned with the hyper-parameters), so it performs a *weighted average* of the past observations. There is *1 free parameter* and *2 hyper-parameters* (trained as described in the previous sections).

- **Hyper Linear 1** model: this is like the **Linear 1** model except that there are weights on the past data (learned with the hyper-parameters), so it performs a *weighted linear regression* over the past input/output pairs. There are *2 free parameters* and *2 hyper-parameters* (trained as described in the previous sections).

	MSE	(sdev)	p-value
Constant	**7.779e-3**	(2.02e-4)	
Hyper Constant	8.075e-3	(2.13e-4)	**W <1e-7**
Hyper Linear 1	9.484e-3	(5.20e-4)	**W1.35e-5**
Linear 1	7.884e-3	(2.08e-4)	**W <1e-7**
ARMA(1,1)	7.908e-3	(2.12e-4)	**W <1e-7**

Table 3.1: Results of experiments on predicting one-month ahead **stocks returns** using a variety of models. The average out-of-sample squared error times 0.5 (MSE) over all the assets are given, with estimated standard deviation of the average in parentheses, and p-value of the null hypothesis of no difference with the Constant model. A "W" means that the alternative hypothesis is that the model is WORSE than the constant model. *All the models are significantly worse than the **Constant** model.*

3.5.3 Experimental Results

The results on predicting the first moment of next month's stocks returns are given in table 3.1. Note that the p-values are one-sided but that we do two different tests depending on whether the tested model average error is less or greater than the reference (Constant model). In the latter case there is a "W" in the p-value column, indicating that the performance is worse than the reference. In that case a low p-value allows to reject the the hypothesis that the constant model is not better than the tested model (i.e., we accept that the tested model is signicantly worse than the constant model). Significant results at the 5% level are indicated with a bold p-value. The lowest error over all the models is indicated by a bold MSE. Note that the constant model significantly beats all the others. Similarly, the results on predicting the second moment are given in table 3.2. Note that the constant model with hyper-parameters significantly beats all the others.

	MSE	(sdev)	p-value
Constant	3.282e-3	(2.02e-4)	
Hyper Constant	**3.215e-3**	(2.13e-4)	**0.0105**
Linear 1	5.208e-3	(1.87e-3)	W0.117
Linear 2	5.707e-3	(1.82e-3)	W0.0522
Linear 4	5.342e-3	(1.89e-3)	W0.0994
ARMA(1,1)	5.617e-3	(2.22e-3)	**W1.94e-3**
ARMA(2,1)	5.515e-3	(2.08e-3)	**W1.79e-3**
ARMA(1,2)	5.910e-3	(2.30e-3)	**W2.01e-3**
ARMA(2,2)	5.637e-3	(1.97e-3)	**W1.66e-3**

Table 3.2: Results of experiments on predicting one-month ahead **stocks squared returns** using a variety of models. The average out-of-sample squared error times 0.5 (MSE) over all the assets are given, with estimated standard deviation of the average in parentheses, and p-value of the null hypothesis of no difference with the Constant model. A "W" means that the alternative hypothesis is that the model is WORSE than the constant model. *The **Hyper Constant** model is significantly better than all the others while the **ARMA** models are all significantly worse than the Constant reference.*

Similar tables (3.3 and 3.4) give the results on predicting respectively the first and second moments of the returns of the 14 sectors (sub-portfolios) of the TSE300 portfolio, while tables 3.5 and 3.6 give the results on the TSE300 market index (i.e., in this case there is only one asset, so the variances of the averages are larger).

3.6 Conclusions

In this paper we have achieved the following:

- We have introduced an extension of cross-validation, called **sequential cross-validation** as a model selection criterion for possibly non-*i.i.d.* data.

- We have applied the method for optimizing hyper-parameters introduced in (7) to the special case of capturing abrupt changes in non-

	MSE	(sdev)	p-value
Constant	1.917e-3	(7.81e-5)	
Hyper Constant	1.927e-3	(7.81e-5)	**W3.97e-2**
Hyper Linear 1	1.927e-3	(7.88e-5)	W0.186
Linear 1	**1.909e-3**	(7.76e-5)	0.128
ARMA(1,1)	1.914e-3	(7.84e-5)	0.228

Table 3.3: Results of experiments on predicting one-month ahead **sector returns** using a variety of models. The average out-of-sample squared error times 0.5 (MSE) over all the assets are given, with estimated standard deviation of the average in parentheses, and p-value of the null hypothesis of no difference with the Constant model. A "W" means that the alternative hypothesis is that the model is WORSE than the constant model. *No model is significantly better than the* ***Constant*** *model.*

stationary data: 2 hyper-parameters control the weight on each past time step.

- We have tested the method on artificial data, showing that when there is such an abrupt change, the regression is much improved by using and optimizing the hyper-parameters.
- We have tested the method on financial returns data to predict the first and second moment of next month's return: for individual stocks, sector aggregates, and a market index. The specific conclusions of these experiments are the following:
  - On estimating the conditional expectation of the return of individual stocks and the market index, the constant model significantly beats all the tested models, including linear and ARMA models. In the case of sectors, none of the models beats significantly the constant model, while the constant model significantly beats some of them.
  - On estimating the conditional expectation of the squared return (which can be used to predict volatility), the constant model with hyper-parameters to handle non-stationarities always beats the other models, but the measurable significance decreases as the

	MSE	(sdev)	p-value
Constant	3.293e-5	(4.07e-6)	
Hyper Constant	**3.242e-5**	(3.76e-6)	0.138
Hyper Linear 1	3.954e-5	(8.18e-6)	W0.153
Linear 1	3.290e-5	(4.01e-6)	0.421
ARMA(1,1)	3.295e-5	(4.05e-6)	W0.425

Table 3.4: Results of experiments on predicting one-month ahead **sectors squared returns** using a variety of models. The average out-of-sample squared error times 0.5 (MSE) over all the assets are given, with estimated standard deviation of the average in parentheses, and p-value of the null hypothesis of no difference with the Constant model. A "W" means that the alternative hypothesis is that the model is WORSE than the constant model. *The **Hyper-Constant** model performs best but the p-value of 13.8% is not very significant, and there are no significant differences one way or another for any of the models with respect to the **Constant** model.*

number of assets is decreased. In the case of the 473 stocks, we obtain a p-value of 1%, in the case of the 14 sectors, 13.8%, and for the single market index, 39.7%. The constant model still beats significantly the ARMA models in all cases.

What remains to be done, in the direction of research that we have explored here? In our experiments we have found that the estimates of the hyper-parameters was very sensitive to the data, probably because of the variance of the cross-validation criterion, so better results might be obtained by using a less noisy model selection criterion.

It would also be interesting to see if the significant improvements that we have found for Canadian stocks can be observed on other markets, and if these predictions could be used to improve specific decisions concerning those stocks (such as for trading options).

	MSE	(sdev)	p-value
Constant	1.009e-3	(1.40e-4)	
Hyper Constant	**1.008e-3**	(1.41e-4)	0.176
Hyper Linear 1	1.012e-3	(1.42e-4)	W0.329
Linear 1	1.012e-3	(1.42e-4)	W0.337
ARMA(1,1)	1.016e-3	(1.42e-4)	W0.316

Table 3.5: Results of experiments on predicting one-month ahead **market index returns** using a variety of models. The average out-of-sample squared error times 0.5 (MSE) over all the assets are given, with estimated standard deviation of the average in parentheses, and p-value of the null hypothesis of no difference with the Constant model. A "W" means that the alternative hypothesis is that the model is WORSE than the constant model. *No model is significantly better or worse than the **Constant** model.*

	MSE	(sdev)	p-value
Constant	8.719e-6	(3.18e-6)	
Hyper Constant	**8.683e-6**	(3.09e-6)	0.397
Hyper Linear 1	8.786e-6	(3.11e-6)	W0.278
Linear 1	8.789e-6	(3.19e-6)	W0.170
ARMA(1,1)	8.845e-6	(3.20e-6)	**W0.035**

Table 3.6: Results of experiments on predicting one-month ahead **market index squared returns** using a variety of models. The average out-of-sample squared error times 0.5 (MSE) over all the assets are given, with estimated standard deviation of the average in parentheses, and p-value of the null hypothesis of no difference with the Constant model. A "W" means that the alternative hypothesis is that the model is WORSE than the constant model. *No model is significantly better or worse than the **Constant model**, except the **ARMA(1,1)** which is significantly worse.*

Chapitre 4

Synthèse

Dans l'article précédent, nous avons obtenus les résultats suivants :

- Nous avons introduit une extension de la validation croisée, appellée validation séquentielle comme critère de sélection de modèle pour des données possiblement non i.i.d.
- Nous avons appliqué la méthode introduite dans (7) pour optimiser les hyper-paramètres dans le cas spécial où deux hyper-paramètres contrôlent l'importance accordée à chacune des données du passé, permettant ainsi de tenir compte de la présence de non-stationnarités dans les données.
- Nous avons testé la méthode sur des données artificielles, montrant que lorsque de telles non-stationarités sont présentes, la régression est largement améliorée par l'utilisation et l'optimisation d'hyper-paramètres.
- Nous avons testé la méthode pour la prédiction des premier et second moment des rendements des actions canadiennes, indices sectoriels et indice de marché. Les conclusions spécifiques à ces expériences sont les suivantes :
  - Pour l'estimation de l'espérance conditionnelle des rendements, le modèle constant a performé de façon significativement meilleure que les autres modèles, incluant les modèles linéaires et ARMA. Dans le cas des données sectorielles, aucun des modèles n'a pu battre le modèle constant de façon significative. À l'inverse, le modèle constant a pu battre de façon significative certains autres modèles considérés.
  - Pour l'estimation de l'espérance conditionnelle du second moment des rendements (rendements au carré qui peuvent servir à la prédiction de la volatilité), le modèle constant avec hyper-paramètres bat

toujours les autres modèles, mais le niveau de signifiance décroît avec le nombre de titres considérés. On obtient un niveau de signifiance de 1% pour les 473 actions, 13.8% pour les données des 14 secteurs et 39.7% pour l'indice de marché TSE300. Notons que le modèle constant bat significativement les modèles ARMA dans tous les cas.

Suite aux expériences réalisées, de nombreuses pistes de recherche s'ouvrent. Nos expériences nous ont montré que les valeurs des hyper-paramètres sont très sensibles aux données, probablement à cause de la variance du critère de validation croisée et donc de meilleurs résultats pourraient être obtenus avec un critère de sélection de modèle moins bruité.

Il serait aussi intéressant de voir si les améliorations statistiquement significatives observées sur les données canadiennes peuvent être obtenues sur d'autres marchés et si les prédictions réalisées pourraient servir à améliorer la prise de certaines décisions sur les titres financiers considérés.

Deuxième partie

Deuxième article

Chapitre 5

Introduction

L'incorporation, dans l'architecture d'un algorithme d'apprentissage, de connaissances *a priori* reliées à une tâche spécifique, peut permettre d'améliorer la performance de généralisation. Dans l'article qui suit, nous étudions le cas particulier d'une fonction positive, dont les dérivées premières par rapport aux deux variables d'entrée sont positives et enfin, dont la dérivée seconde par rapport à l'une des deux variables est, elle aussi, positive. À cette fin, nous proposons une nouvelle classe de fonctions, semblables aux réseaux de neurones multi-couches, qui possèdent les propriétés énoncées. La classe définie est un approximateur universel pour les fonctions continues possédant ces propriétés et certaines autres propriétés. Nous appliquons cette nouvelle classe de fonctions à la valorisation d'options d'achat. Les expériences réalisées démontrent une amélioration des performance de régression par le biais de l'utilisation de la classe de fonctions proposée.

Il a déjà été démontré que la classe de fonctions des réseaux de neurones à une couche cachée est un approximateur universel pour les fonctions continues (23; 24; 50; 5). De plus, (64) ont montré que toute fonction de transfert non polynômiale, utilisée dans les unités cachées, permettra d'atteindre ce but. L'imposition de contraintes sur la classe de fonctions réduit l'espace considéré et de ce fait, on s'attend à ce que la variance en soit d'autant réduite, ce qui devrait nous mener vers un nouvel équilibre dans la résolution du dilemme biais-variance (39; 68). Cette hypothèse est confirmée par nos expériences réalisées sur des données artificielles.

Le volume de transactions sur les options financières connut un important essor lorsque (11) publièrent leur formule, relativement simple, pour la valorisation de certains de ces produits financiers. Malheureusement, cette

formule se fait forte de plusieurs hypothèses paramétriques, contredites par l'analyse empirique (12). De plus, toute erreur dans l'énoncé du processus stochastique de l'actif sous-jacent d'une option financière se traduit par l'introduction d'un biais systématique dans la valorisation de cette option (59). Dans ce contexte, les algorithmes d'apprentissage, libres d'hypothèses paramétriques, peuvent potentiellement apporter une amélioration au niveau prédictif. Enfin, (38) ont proposé une architecture de réseaux de neurones basée sur la formule de Black-Scholes, ce qui correspond à une autre façon d'incorporer des connaissances *a priori*.

L'auteur de cette thèse s'est basé sur la fonction « softplus » et l'existence d'une preuve d'universalité pour le cas unidimensionel. La contribution de l'auteur a été de concevoir l'architecture multi-dimensionnelle, développer la preuve d'universalité multi-dimensionelle, concevoir la preuve d'universalité pour l'utilisation de la fonction softplus en sortie, réaliser les expériences numériques, rédiger les articles (29; 30) et effectuer une présentation orale lors de la conférence NIPS*2001.

Chapter 6

Incorporating Second-Order Functional Knowledge for Better Option Pricing

Incorporating prior knowledge of a particular task into the architecture of a learning algorithm can greatly improve generalization performance. We study here a case where we know that the function to be learned is non-decreasing in its two arguments and convex in one of them. For this purpose we propose a class of functions similar to multi-layer neural networks but (1) that has those properties, (2) is a universal approximator of Lipschitz[1] functions with these and other properties. We apply this new class of functions to the task of modeling the price of call options. Experiments show improvements on regressing the price of call options using the new types of function classes that incorporate the *a priori* constraints.

6.1 Introduction

Incorporating *a priori* knowledge of a particular task into a learning algorithm helps to reduce the necessary complexity of the learner and generally improves performance, if the incorporated knowledge is relevant to the task and brings enough information about the unknown generating process of the data. In this paper we consider prior knowledge on the positivity of some first and second derivatives of the function to be learned. In particular such

[1] A function f is Lipschitz in Ω if $\exists c > 0,\ \forall x, y \in \Omega,\ |f(y) - f(x)| \leq c|y - x|$.(25)

constraints have applications to modeling the price of stock options. Based on the Black-Scholes formula, the price of a call stock option is monotonically increasing in both the "moneyness" and time to maturity of the option, and it is convex in the "moneyness". Section 6.4 better explains these terms and stock options. For a function $f(x_1, x_2)$ of two real-valued arguments, this corresponds to the following properties:

$$f \geq 0, \qquad \frac{\partial f}{\partial x_1} \geq 0, \qquad \frac{\partial f}{\partial x_2} \geq 0, \qquad \frac{\partial^2 f}{\partial x_1^2} \geq 0 \tag{6.1}$$

The mathematical results of this paper (section 2) are the following: first we introduce a class of one-argument functions that is positive, non-decreasing and convex in its argument. Second, we use this new class of functions as a building block to design another class of functions that is a universal approximator for functions with positive outputs. Third, once again using the first class of functions, we design a third class that is a universal approximator to functions of two or more arguments, with the set of arguments partitioned in two groups: those arguments for which the second derivative is known positive and those arguments for which we have no prior knowledge on the second derivative. The first derivative is positive for any argument. The universality property of the third class rests on additional constraints on cross-derivatives, which we illustrate below for the case of two arguments:

$$\frac{\partial^2 f}{\partial x_1 \partial x_2} \geq 0, \quad \frac{\partial^3 f}{\partial x_1^2 \partial x_2} \geq 0 \tag{6.2}$$

Thus, we assume that $f \in C^3$, the set of functions three times continuously differentiable. Comparative experiments on these new classes of functions were performed on stock option prices, showing improvements when using these new classes rather than ordinary feedforward neural networks. The improvements appear to be non-stationary but the new class of functions shows the most stable behavior in predicting future prices. The detailed experimental results are presented in section 6.6.

6.2 Theory

Definition 6.2.1 *A class of functions $\hat{\mathcal{F}}$ from $\mathbb{R}^n$ to $\mathbb{R}$ is a* **universal approximator** *for a class of functions $\mathcal{F}$ from $\mathbb{R}^n$ to $\mathbb{R}$ if for any $f \in \mathcal{F}$, any*

compact domain $D \subset \mathbb{R}^n$, *and any positive* ϵ, *one can find a* $\hat{f} \in \hat{\mathcal{F}}$ *with* $\sup_{x \in D} |f(x) - \hat{f}(x)| \leq \epsilon$.

It has already been shown that the class of artificial neural networks with one hidden layer

$$\hat{\mathcal{N}} = \{f(x) = b_0 + \sum_{i=1}^{H} w_i h(b_i + \sum_j v_{ij} x_j)\} \tag{6.3}$$

e.g. with a sigmoid activation function $h(s) = 1/(1 + e^{-s})$, is a universal approximator of continuous functions (23; 24; 50; 5). Furthermore,(64) have shown that any non-polynomial activation function will suffice for universal approximation. The number of hidden units H of the neural network is a hyper-parameter that controls the accuracy of the approximation and it should be chosen to balance the trade-off between accuracy (bias of the class of functions) and variance (due to the finite sample used to estimate the parameters of the model), see also (68). Because of this trade-off, in the finite sample case, it may be advantageous to consider a "simpler" class of functions that is appropriate to the task.

Since the sigmoid h is monotonically increasing, it is easy to force the first derivatives with respect to x to be positive by forcing the weights to be positive, for example with the exponential function:

$$\hat{\mathcal{N}}_+ = \{f(x) = e^{b_0} + \sum_{i=1}^{H} e^{w_i} h(b_i + \sum_j e^{v_{ij}} x_j)\} \tag{6.4}$$

because $h'(s) = h(s)(1-h(s)) > 0$. Note that the positivity of $f(x)$ and $f'(x)$ is not affected by the values of the $\{b_i\}$ parameters. Since the sigmoid h has a positive first derivative, its primitive, which we call *softplus*, is convex:

$$\boxed{\zeta(s) = \ln(1 + e^s)} \tag{6.5}$$

where $\ln(\cdot)$ is the natural logarithm operator. Note that $d\zeta(s)/ds = h(s) = 1/(1 + e^{-s})$.

6.2.1 Universality for functions with positive outputs

Using the softplus function introduced above, we define a new class of functions, all of which have positive outputs:

$$\hat{\mathcal{N}}_{>0} = \{f(x) = \zeta(g(x)), g(x) \in \hat{\mathcal{N}}\} \tag{6.6}$$

Theorem 6.2.1 *Within the set of continuous functions from* $\mathbb{R}^n$ *to* $\mathbb{R}_+ = \{x : x \in \mathbb{R}, x > 0\}$, *the class* $\hat{\mathcal{N}}_{>0}$ *is a universal approximator.*

Proof Consider a positive function $f(x)$, which we want to approximate arbitrarily well. Consider $g(x) = \zeta^{-1}(f(x)) = \ln(e^{f(x)} - 1)$, the inverse softplus transform of $f(x)$. Choose $\hat{g}(x)$ from $\hat{\mathcal{N}}$ such that $\sup_{x \in D} |g(x) - \hat{g}(x)| \leq \epsilon$, where D is any compact domain over $\mathbb{R}^n$ and ϵ is any positive real number. The existence of $\hat{g}(x)$ is ensured by the universality property of $\hat{\mathcal{N}}$. Set $\hat{f}(x) = \zeta(\hat{g}(x)) = \ln(1 + e^{\hat{g}(x)})$. Consider any particular x and define $a = \min(\hat{g}(x), g(x))$ and $b = \max(\hat{g}(x), g(x))$. Since $b - a \leq \epsilon$, we have,

$$\begin{aligned} |\hat{f}(x) - f(x)| &= \ln(1 + e^b) - \ln(1 + e^a) \\ &\leq \ln\left(1 + (e^\epsilon - 1)e^a/(1 + e^a)\right) \\ &< \epsilon \end{aligned}$$

and the proof is complete. Thus, the use of the softplus function to transform the output of a regular one hidden layer artificial neural network ensures the positivity of the final output without hindering the universality property.

6.2.2 The class ${}_{c,n}\hat{\mathcal{N}}_{++}$

In this section, we use the softplus function, in order to define a new class of functions with positive outputs, positive first derivatives w.r.t. all input variables and positive second derivatives w.r.t. some of the input variables. The basic idea is to replace the sigmoid of a sum by a product of either softplus or sigmoid functions over each of the dimensions (using the softplus over the convex dimensions and the sigmoid over the others):

$${}_{c,n}\hat{\mathcal{N}}_{++} = \{f(x) = e^{b_0} + \sum_{i=1}^{H} e^{w_i} (\underbrace{\prod_{j=1}^{c} \zeta(b_{ij} + e^{v_{ij}} x_j)}_{\text{convex dimensions}} \underbrace{(\prod_{j=c+1}^{n} h(b_{ij} + e^{v_{ij}} x_j))}_{\text{non-convex dimensions}})\} \tag{6.7}$$

One can readily check that the output is necessarily positive, the first derivatives w.r.t. x_j are positive, and the second derivatives w.r.t. x_j for $j \leq c$ are positive. However, this class of functions has other properties as well. Let $(j_1, \cdots, j_m)$ be a set of indices with $1 \leq j_i \leq c$ (convex dimensions), and let $(j'_1, \cdots, j'_p)$ be a set of indices $c+1 \leq j'_i \leq n$ (the other dimensions), then

$$\frac{\partial^{m+p} f}{\partial x_{j_1} \cdots \partial x_{j_m} \partial x_{j'_1} \cdots x_{j'_p}} \geq 0, \quad \frac{\partial^{2m+p} f}{\partial x_{j_1}^2 \cdots \partial x_{j_m}^2 \partial x_{j'_1} \cdots x_{j'_p}} \geq 0 \tag{6.8}$$

where we have assumed that $f \in \mathcal{C}^{c+n}$, the set of functions that are $c+n$ times continuously differentiable. We will also restrict ourselves to Lispschitz functions since the proof of the theorem relies on the fact that the derivative of the function is bounded. The set of functions that respect these derivative conditions will be referred to as ${}_{c,n}\mathcal{F}_{++}$. Note that m or p can be 0, so as special cases we find that f is positive, and that it is monotonically increasing w.r.t. all its inputs, and convex w.r.t. the first c inputs. Also note that this set of equations, when applied to our particular case where $n = 2, c = 1$, corresponds to equations 6.1 and 6.2.

6.2.3 Universality of ${}_{c,n}\hat{\mathcal{N}}_{++}$

We now state the main universality theorem:

Theorem 6.2.2 *Within the set ${}_{c,n}\mathcal{F}_{++}$ of Lipschitz functions from $\mathbb{R}^n$ to $\mathbb{R}$ whose set of derivatives as specified by equation 6.8 are non-negative, the class ${}_{c,n}\hat{\mathcal{N}}_{++}$ is a universal approximator.*

The proof of the theorem is given in section 6.8.

6.3 Experiments with Artificial Data

In this section, we present a series of controlled experiments in order to assess the potential improvements that can be gained from using the proposed architecture in cases where some derivatives of the target function are known to be positive. The emphasis is put on analyzing the evolution of the model bias and model variance values w.r.t. various noise levels.

The function we shall attempt to learn is

$$f(\vec{x}) = \zeta(x_1)\zeta(x_2)\zeta(x_3)h(x_4) \tag{6.9}$$

$$y = f(\vec{x}) + \xi \tag{6.10}$$

where $\zeta(\cdot)$ is the softplus function defined above and $h(\cdot)$ is the sigmoid function. The input values are drawn from a uniform distribution over the [0,1] interval, i.e., $x_i \sim \mathcal{U}(0,1)$. The noise term ξ is added to the true function $f(\vec{x})$ to generate the target value y. Finally, $\xi \sim \mathcal{N}(0,\sigma^2)$, i.e., we used additive gaussian noise. Different values for σ^2 have been tested.

In a first stage of model selection, we trained one model for each combination of number of hidden units, learning rate, learning rate decrease and weight decay. In order to select the hyper-parameters, we used a validation set of the same size as the training set. Once the model parameters were chosen, we trained 10 models using different training and validation datasets (using different random seeds) of the same size as the initial training and validation datasets. The five models with the best validation performance were kept. Bias and variance were measured using these five selected models when applied on another testset of 10000 examples. In each case, the number of training epochs was 10000.

In order to compute the bias and variance values, we first computed, for each test example, the average of the five model outputs:

$$\bar{f}(\vec{x}_i) = \frac{1}{N_D} \sum_D \hat{f}_D(\vec{x}_i). \tag{6.11}$$

The bias was unbiasedly estimated as the average over all test examples, of the squared deviation of the mean output $\bar{f}(\vec{x}_i)$ w.r.t. the known true function value $f(\vec{x}_i)$:

$$b = \frac{1}{N_i} \sum_i (\bar{f}(\vec{x}_i) - f(\vec{x}_i))^2. \tag{6.12}$$

The variance was unbiasedly approximated as the average over all test examples, of the sample variance of model outputs $\hat{f}_D(\vec{x}_i)$ w.r.t. the corresponding mean output $\bar{f}(\vec{x}_i)$:

$$v = \frac{1}{N_i(N_D - 1)} \sum_i \sum_D (\hat{f}_D(\vec{x}_i) - \bar{f}(\vec{x}_i))^2 \tag{6.13}$$

Bias and Variance Analysis on Artificial Data

Ntrain	Noise	Architecture	Bias	Variance	Sum
50	1e-04	Ann	5.29e-05	12.50e-05	17.79e-05
		Convex	2.10e-05	1.78e-05	3.88e-05
	1e-02	Ann	21.26e-04	67.83e-04	89.09e-04
		Convex	5.02e-04	28.91e-04	33.93e-04
	4e-02	Ann	5.28e-03	12.91e-03	18.19e-03
		Convex	1.44e-03	1.17e-03	2.62e-03
100	1e-04	Ann	1.72e-05	3.06e-05	4.77e-05
		Convex	1.12e-05	3.28e-05	4.39e-05
	1e-02	Ann	8.49e-04	16.36e-04	24.85e-04
		Convex	2.90e-04	4.17e-04	7.08e-04
	4e-02	Ann	2.43e-03	5.19e-03	7.61e-03
		Convex	1.35e-03	2.91e-03	4.26e-03
200	1e-04	Ann	1.05e-05	1.19e-05	2.23e-05
		Convex	0.77e-05	0.81e-05	1.58e-05
	1e-02	Ann	2.43e-04	6.52e-04	8.96e-04
		Convex	5.40e-04	4.76e-04	10.16e-04
	4e-02	Ann	0.58e-03	1.48e-03	2.06e-03
		Convex	0.35e-03	1.36e-03	1.71e-03

Table 6.1: Comparison of the bias and variance values for two neural network architectures, three levels of noise, and three sizes of training sets (Ntrain), using artificial data.

Table 1 reports the results for these simulations. For all but one combination of noise level and sample size, the bias and variance are lower for the proposed architecture than for a regular neural network architecture, which is the result we expected. The variance reduction is easy to understand because of the appropriate constraints on the class of functions. The bias reduction, we conjecture to be a side effect of the bias-variance tradeoff being performed by the model selection on the validation set: to achieve a lower validation error, a larger bias is needed with the unconstrained artificial neural network. The improvements are generally more important for smaller sample sizes. A possible explanation is that the proposed architecture helps reduce the variance of the estimator. With small sample sizes, this is very beneficial and becomes less important as the number of points increases.

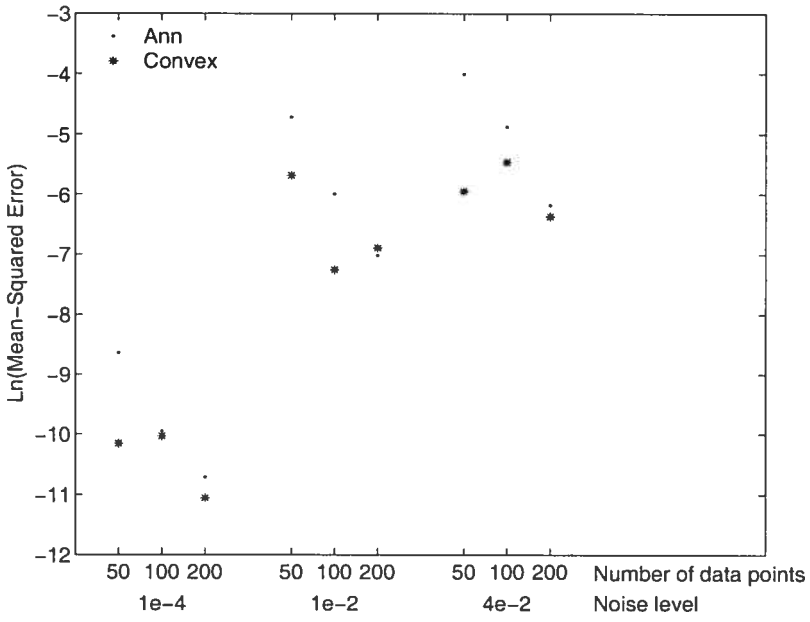

Figure 6.1: *Logarithm of mean-squared error on artificial data experiments. We compare a regular artificial neural network with the proposed architecture using 50, 100 and 200 points and with noise levels of 1e-4, 1e-2 and 4e-2.*

6.4 Estimating Call Option Prices

An option is a contract between two parties that entitles the buyer to a claim at a future date T that depends on the future price, S_T of an underlying asset whose price at current time t is S_t. In this paper we consider the very common European call options, in which the buyer (holder) of the option obtains the right to buy the asset at a fixed price K called the strike price. This purchase can only occur at maturity date (time T). Thus, if at maturity, the price of the asset S_T is above the strike price K, the holder of the option can *exercise* his option and buy the asset at price K, then sell it back on the market at price S_T, thus making a profit of $S_T - K$. If, on the other hand, the price of the asset at maturity S_T is below the strike price K, then the holder of the option has no interest in exercising his option (and does not have to) and the option simply expires worthless and unexercised. For this reason, the option is considered to be worth $\max(0, S_T - K)$ at maturity and our goal is to estimate the value of that worth at current time t.

In the econometric litterature, the call function is often expressed in terms of the primary economic variables that influence its value: the actual market price of the security (S_t), the strike price (K), the remaining time to maturity ($\tau = T - t$), the risk free interest rate (r), and the volatility of the return (σ). One important result is that under mild conditions, the call option function is homogeneous of degree one with respect to the strike price and so we can perform dimensionality reduction by letting our approximating function depend on the "moneyness" ratio ($M = S_t/K$) instead of the current asset price S_t and the strike price K independently. We must then modify the target to be the price of the option divided by the strike price: C_t/K.

Most of the research on call option modelling relies on strong parametric assumptions of the underlying asset price dynamics. Any misspecification of the stochastic process for the asset price will lead to systematic mispricings for any option based on the asset (59). The well-known Black-Scholes formula (11) is a consequence of such specifications and other assumptions:

$$f(M, \tau, r, \sigma) \quad = \quad M\mathcal{N}(d_1) - e^{-r\tau}\mathcal{N}(d_2) \tag{6.14}$$

where $\mathcal{N}(\cdot)$ is the cumulative gaussian function evaluated in points

$$d_1, d_2 \quad = \quad \frac{\ln M + (r \pm \sigma^2/2)\tau}{\sigma\sqrt{\tau}} \tag{6.15}$$

i.e., $d_1 = d_2 + \sigma\sqrt{\tau}$. Two assumptions on which this formula relies have been challenged by empirical evidence: the assumed lognormality of returns on the asset and the assumed constance of volatility over time.

On the other hand, nonparametric models such as neural networks do not rely on such strong assumptions and are therefore robust to model specification errors and their consequences on option modelling and this motivates research in the direction of applying nonparametric techniques for option modelling.

Analyzing the primary economic variables that influence the call option price, we note that the risk free interest rate (r) needs to be somehow extracted from the term structure of interest rates and the volatility (σ) needs to be forecasted. This latter task is a field of research in itself. We have (28) previously tried to feed in neural networks with estimates of the volatility using historical averages but so far, the gains remained insignificant. We therefore drop these two features and rely on the ones that can be observed (S_t, K, τ) to obtain the following:

$$C_t/K = f(M, \tau). \tag{6.16}$$

The novelty of our approach is to account for properties of the call option function as stated in equation 6.1. These properties derive from simple arbitrage pricing theory[2]. Now even though we know the call option function to respect these properties, we do not know if it does respect the additional cross derivative properties of equation 6.2. In order to gain some insight in this direction, we confront the Black-Scholes formula to our set of constraints:

$$\frac{\partial f}{\partial M} = \mathcal{N}(d_1) \tag{6.17}$$

$$\frac{\partial^2 f}{\partial M^2} = \frac{\mathcal{N}'(d_1)}{\sqrt{\tau} M \sigma} \tag{6.18}$$

$$\frac{\partial f}{\partial \tau} = e^{-r\tau}\left(\frac{\mathcal{N}'(d_2)\sigma}{2\sqrt{\tau}} + r\mathcal{N}(d_2)\right) \tag{6.19}$$

$$\frac{\partial^2 f}{\partial M \partial \tau} = \frac{\mathcal{N}'(d_1)}{2\sigma\tau^{3/2}}\left((r + \sigma^2/2)\tau - \ln M\right) \tag{6.20}$$

$$\frac{\partial^3 f}{\partial M^2 \partial \tau} = \frac{\mathcal{N}'(d_1)}{2M\sigma^3\tau^{5/2}}\left(\ln^2 M - \sigma^2\tau - (r + \sigma^2/2)^2\tau^2\right) \tag{6.21}$$

[2]The convexity of the call option w.r.t. the moneyness is a consequence of the butterfly spread strategy.(38)

where $\mathcal{N}'(\cdot)$ is the gaussian density function. Equations 6.17, 6.18 and 6.19 confirm that the Black-Scholes formula is in accordance with our prior knowledge of the call option function: all three derivatives are positive. Equations 6.20 and 6.21 are the cross derivatives which will be positive for any function chosen from ${}_{1,2}\hat{\mathcal{N}}_{++}$. When applied to the Black-Scholes formula, it is less clear whether these values are positive, too. In particular, one can easily see that both cross derivatives can not be simultaneously positive. Thus, the Black-Scholes formula is not within the set ${}_{1,2}\hat{\mathcal{F}}_{++}$. Then again, it is known that the Black-Scholes formula does not adequately represent the market pricing of options, but it is considered as a useful guide for evaluating call option prices. So, we do not know if these constraints on the cross derivatives are present in the true price function.

Nonetheless, even if these additional constraints are not respected by the true function on all of its domain, one can hope that the increase in the bias of the estimator due to the constraints will be offset (because we are searching in a smaller function space) by a decrease in the variance of that estimator and that overall, the mean-squared error will decrease. This strategy has often been used successfully in machine learning (e.g., regularization, feature selection, smoothing, etc.).

6.5 Experimental Setup

As a reference model, we use a simple multi-layered perceptron with one hidden layer (eq. 6.3). We also compare our results with a recently proposed model (38) that closely resembles the Black-Scholes formula for option pricing (i.e. another way to incorporate possibly useful prior knowledge):

$$\begin{aligned} y &= \alpha + M \cdot \sum_{i=1}^{n_h} \beta_{1,i} \cdot h(\gamma_{i,0} + \gamma_{i,1} \cdot M + \gamma_{i,2} \cdot \tau) \\ &+ e^{-r\tau} \cdot \sum_{i=1}^{n_h} \beta_{2,i} \cdot h(\gamma_{i,3} + \gamma_{i,4} \cdot M + \gamma_{i,5} \cdot \tau), \end{aligned} \tag{6.22}$$

with inputs M, τ, parameters r, α, β, γ and hyperparameters n_h (number of hidden units). We shall refer to equation 6.22 as the Garcia model. We evaluate two new architectures incorporating all of the constraints defined in equation 6.8. The proposed class is referred to as "Products of Softplus and Sigmoid functions". We also tested another architecture derived from

the proposed one by simply summing, instead of multiplying, softplus and sigmoid functions. For that last architecture, all constraints on positivity, monotonicity and convexity are respected but in that case, cross-derivatives are all equal to zero. We do not have a universality proof for that specific class of functions. We refer to models within that class of functions as "Sums of Softplus and Sigmoid functions".

We used European call option data from 1988 to 1993. A total of 43518 transaction prices on European call options on the S&P500 index were used. In figures 6.2,6.3 and 6.4, data for years 1988, 1989 and 1990 are plotted. In section 6.6, we report results on 1988 data. In each case, we used the first two quarters of 1988 as a training set (3434 examples), the third quarter as a validation set (1642 examples) for model selection and the fourth quarter as a test set (each with around 1500 examples) for final generalization error estimation. In tables 6.2 and 6.3, we present results for networks with unconstrained weights on the left-hand side, and weights constrained to positive and monotone functions through exponentiation of parameters on the right-hand side. For each model, the number of hidden units varies from one to nine. The mean squared error results reported were obtained as follows: first, we randomly sampled the parameter space 1000 times. We picked the best (lowest training error) model and trained it up to 1000 more epochs. Repeating this procedure 10 times, we selected and averaged the performance of the best of these 10 models (those with training error no more than 10% worse than the best out of 10). In figures 6.5 and 6.6, we present tests of the same models on each quarter up to and including 1993 (20 additional test sets) in order to assess the persistence (conversely, the degradation through time) of the trained models.

6.6 Forecasting Results

As can be seen in tables 6.2 and 6.3, the unconstrained networks obtain better training, validation and testing (test 1) results but fail in the extra testing set (test 2). A possible explanation is that constrained architectures capture more fundamental relationships between variables and are more robust to nonstationarities of the underlying process. Constrained architectures therefore seem to give better generalization when considering longer time spans.

The importance in the difference in performance between constrained and unconstrained architectures on the second test set lead us to look even

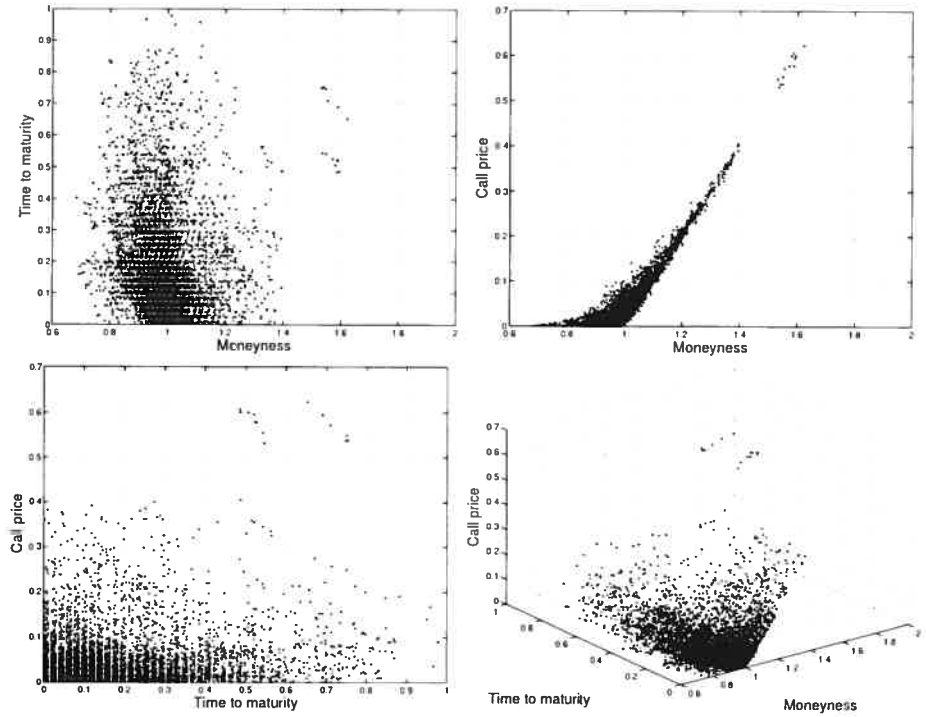

Figure 6.2: *1988 call data. Top left: time to maturity is plotted against moneyness (expressed as the ratio of the spot price to the strike price). Top right: call price against moneyness. Bottom left: call price against time to maturity. Bottom right: three-dimensional plot of call price as a function of both moneyness and time to maturity.*

farther into the future and test the selected models on data from later years. In Figure 6.5 and 6.6, we see that the constrained Black-Scholes similar model performs slightly better than other models on the second test set but then fails on later quarters. All in all, at the expense of slightly higher initial errors our proposed architecture allows one to forecast with increased stability much farther in the future. This is a very welcome property as new derivative products have a tendency to lock in values for much longer durations (up to 10 years) than traditional ones.

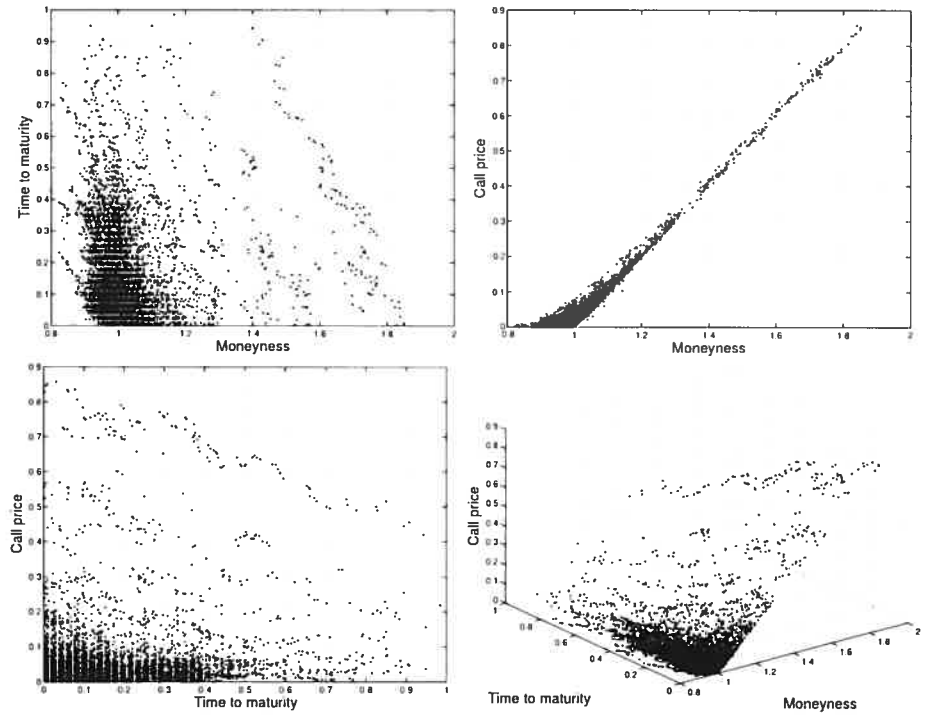

Figure 6.3: *1989 call data. Top left: time to maturity is plotted against moneyness (expressed as the ratio of the spot price to the strike price). Top right: call price against moneyness. Bottom left: call price against time to maturity. Bottom right: three-dimensional plot of call price as a function of both moneyness and time to maturity.*

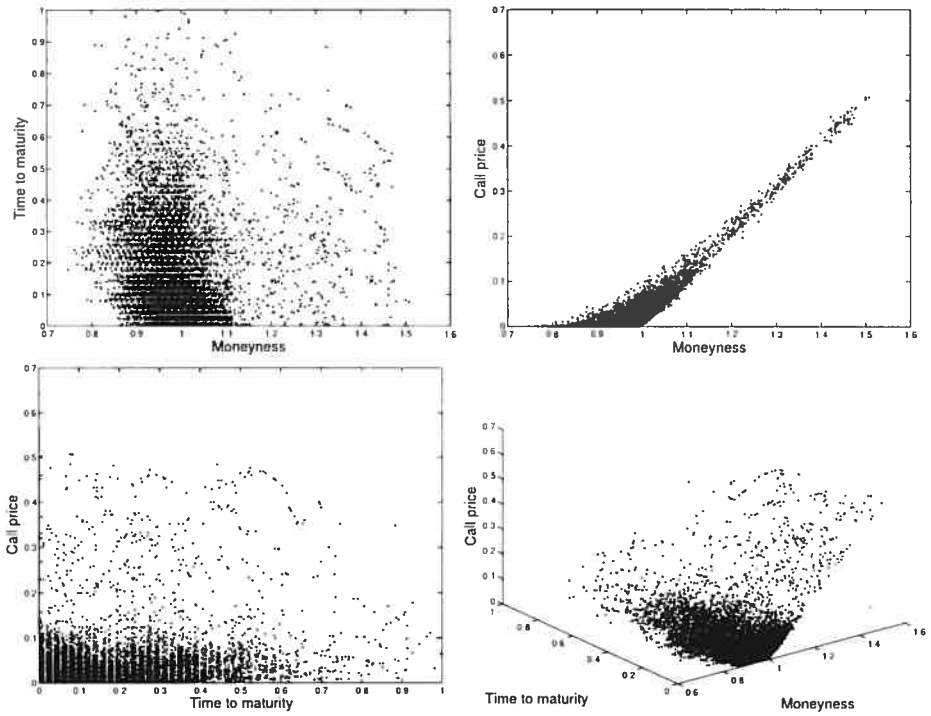

Figure 6.4: *1990 call data. Top left: time to maturity is plotted against moneyness (expressed as the ratio of the spot price to the strike price). Top right: call price against moneyness. Bottom left: call price against time to maturity. Bottom right: three-dimensional plot of call price as a function of both moneyness and time to maturity.*

Simple Multi-Layered Perceptrons

Mean Squared Error Results on Call Option Pricing ($\times 10^{-4}$)

Hidden Units	Unconstrained weights				Constrained weights			
	Train	Valid	Test1	Test2	Train	Valid	Test1	Test2
1	2.38	1.92	2.73	6.06	2.67	2.32	3.02	3.60
2	1.68	1.76	1.51	5.70	2.63	2.14	3.08	3.81
3	1.40	1.39	1.27	27.31	2.63	2.15	3.07	3.79
4	1.42	1.44	1.25	27.32	2.65	2.24	3.05	3.70
5	1.40	1.38	**1.27**	**30.56**	2.67	2.29	3.03	3.64
6	1.41	1.43	1.24	33.12	2.63	2.14	**3.08**	**3.81**
7	1.41	1.41	1.26	33.49	2.65	2.23	3.05	3.71
8	1.41	1.43	1.24	39.72	2.63	2.14	3.07	3.80
9	1.40	1.41	1.24	38.07	2.66	2.27	3.04	3.67

Black-Scholes Similar Networks

Mean Squared Error Results on Call Option Pricing ($\times 10^{-4}$)

Hidden Units	Unconstrained weights				Constrained weights			
	Train	Valid	Test1	Test2	Train	Valid	Test1	Test2
1	1.54	1.58	1.40	4.70	2.49	2.17	2.78	3.61
2	1.42	1.42	1.27	24.53	1.90	1.71	2.05	3.19
3	1.40	1.41	1.24	30.83	1.88	1.73	2.00	3.72
4	1.40	1.39	**1.27**	**31.43**	1.85	1.70	1.96	3.15
5	1.40	1.40	1.25	30.82	1.87	1.70	2.01	3.51
6	1.41	1.42	1.25	35.77	1.89	1.70	2.04	3.19
7	1.40	1.40	1.25	35.97	1.87	1.72	1.98	3.12
8	1.40	1.40	1.25	34.68	1.86	1.69	**1.98**	**3.25**
9	1.42	1.43	1.26	32.65	1.92	1.73	2.08	3.17

Table 6.2: *Left: the parameters are free to take on negative values. Right: parameters are constrained through exponentiation so that the resulting function is both positive and monotone increasing everywhere w.r.t. both inputs as in equation 6.4. Top: regular feedforward artificial neural networks. Bottom: neural networks with an architecture resembling the Black-Scholes formula as defined in equation 6.22. The number of hidden units varies from 1 to 9 for each network architecture. The first two quarters of 1988 were used for training, the third of 1988 for validation and the fourth of 1988 for testing. The first quarter of 1989 was used as a second test set to assess the persistence of the models through time (figures 6.5 and 6.6). In bold: test results for models with best validation results.*

Products of SoftPlus and Sigmoid Functions

Mean Squared Error Results on Call Option Pricing ($\times 10^{-4}$)

Hidden Units	Unconstrained weights				Constrained weights			
	Train	Valid	Test1	Test2	Train	Valid	Test1	Test2
1	2.27	2.15	2.35	3.27	2.28	2.14	2.37	3.51
2	1.61	1.58	1.58	14.24	2.28	2.13	2.37	3.48
3	1.51	1.53	1.38	18.16	2.28	2.13	2.36	3.48
4	1.46	1.51	1.29	20.14	1.84	1.54	**1.97**	**4.19**
5	1.57	1.57	1.46	10.03	1.83	1.56	1.95	4.18
6	1.51	1.53	1.35	22.47	1.85	1.57	1.97	4.09
7	1.62	1.67	1.46	7.78	1.86	1.55	2.00	4.10
8	1.55	1.54	1.44	11.58	1.84	1.55	1.96	4.25
9	1.46	1.47	**1.31**	**26.13**	1.87	1.60	1.97	4.12

Sums of SoftPlus and Sigmoid functions

Mean Squared Error Results on Call Option Pricing ($\times 10^{-4}$)

Hidden Units	Unconstrained weights				Constrained weights			
	Train	Valid	Test1	Test2	Train	Valid	Test1	Test2
1	1.83	1.59	1.93	4.10	2.30	2.19	2.36	3.43
2	1.42	1.45	**1.26**	**25.00**	2.29	2.19	2.34	3.39
3	1.45	1.46	1.32	35.00	1.84	1.58	1.95	4.11
4	1.56	1.69	1.33	21.80	1.85	1.56	1.99	4.09
5	1.60	1.69	1.42	10.11	1.85	1.52	**2.00**	**4.21**
6	1.57	1.66	1.39	14.99	1.86	1.54	2.00	4.12
7	1.61	1.67	1.48	8.00	1.86	1.60	1.98	3.94
8	1.64	1.72	1.48	7.89	1.85	1.54	1.98	4.25
9	1.65	1.70	1.52	6.16	1.84	1.54	1.97	4.25

Table 6.3: *Similar results as in table 6.2 but for two new architectures. Top: products of softplus along the convex axis with sigmoid along the monotone axis. Bottom: the softplus and sigmoid functions are summed instead of being multiplied. Top right: the fully constrained proposed architecture.*

Mean Squared Error Results on Call Option Pricing ($\times 10^{-4}$)

Simple Multi-Layered Perceptrons

Hidden Units	Unconstrained weights				Constrained weights			
	Train	Valid	Test1	Test2	Train	Valid	Test1	Test2
1	2.38	1.92	2.73	6.06	2.67	2.32	3.02	3.60
2	1.68	1.76	1.51	5.70	2.63	2.14	3.08	3.81
3	1.40	1.39	1.27	27.31	2.63	2.15	3.07	3.79
4	1.42	1.44	1.25	27.32	2.65	2.24	3.05	3.70
5	1.40	1.38	**1.27**	**30.56**	2.67	2.29	3.03	3.64
6	1.41	1.43	1.24	33.12	2.63	2.14	**3.08**	**3.81**
7	1.41	1.41	1.26	33.49	2.65	2.23	3.05	3.71
8	1.41	1.43	1.24	39.72	2.63	2.14	3.07	3.80
9	1.40	1.41	1.24	38.07	2.66	2.27	3.04	3.67

Products of SoftPlus and Sigmoid Functions

Unconstrained weights				Constrained weights			
Train	Valid	Test1	Test2	Train	Valid	Test1	Test2
2.27	2.15	2.35	3.27	2.28	2.14	2.37	3.51
1.61	1.58	1.58	14.24	2.28	2.13	2.37	3.48
1.51	1.53	1.38	18.16	2.28	2.13	2.36	3.48
1.46	1.51	1.29	20.14	1.84	1.54	**1.97**	**4.19**
1.57	1.57	1.46	10.03	1.83	1.56	1.95	4.18
1.51	1.53	1.35	22.47	1.85	1.57	1.97	4.09
1.62	1.67	1.46	7.78	1.86	1.55	2.00	4.10
1.55	1.54	1.44	11.58	1.84	1.55	1.96	4.25
1.46	1.47	**1.31**	**26.13**	1.87	1.60	1.97	4.12

Black-Scholes Similar Networks

Hidden Units	Unconstrained weights				Constrained weights			
	Train	Valid	Test1	Test2	Train	Valid	Test1	Test2
1	1.54	1.58	1.40	4.70	2.49	2.17	2.78	3.61
2	1.42	1.42	1.27	24.53	1.90	1.71	2.05	3.19
3	1.40	1.41	1.24	30.83	1.88	1.73	2.00	3.72
4	1.40	1.39	**1.27**	**31.43**	1.85	1.70	1.96	3.15
5	1.40	1.40	1.25	30.82	1.87	1.70	2.01	3.51
6	1.41	1.42	1.25	35.77	1.89	1.70	2.04	3.19
7	1.40	1.40	1.25	35.97	1.87	1.72	1.98	3.12
8	1.40	1.40	1.25	34.68	1.86	1.69	**1.98**	**3.25**
9	1.42	1.43	1.26	32.65	1.92	1.73	2.08	3.17

Sums of SoftPlus and Sigmoid functions

Unconstrained weights				Constrained weights			
Train	Valid	Test1	Test2	Train	Valid	Test1	Test2
1.83	1.59	1.93	4.10	2.30	2.19	2.36	3.43
1.42	1.45	**1.26**	**25.00**	2.29	2.19	2.34	3.39
1.45	1.46	1.32	35.00	1.84	1.58	1.95	4.11
1.56	1.69	1.33	21.80	1.85	1.56	1.99	4.09
1.60	1.69	1.42	10.11	1.85	1.52	**2.00**	**4.21**
1.57	1.66	1.39	14.99	1.86	1.54	2.00	4.12
1.61	1.67	1.48	8.00	1.86	1.60	1.98	3.94
1.64	1.72	1.48	7.89	1.85	1.54	1.98	4.25
1.65	1.70	1.52	6.16	1.84	1.54	1.97	4.25

Table 6.4: *Constrained models use exponentiation of parameters to ensure positivity and monotonicity whereas unconstrained models do not. The number of hidden units varies from 1 to 9 for each network architecture. The first two quarters of 1988 were used for training, the third of 1988 for validation and the fourth of 1988 for testing. The first quarter of 1989 was used as a second test set to assess the persistence of the models through time (see also figures 6.5 and 6.6). In bold: test results for models with best validation results. The product of softplus and sigmoid functions with constrained weights corresponds to the proposed architecture.*

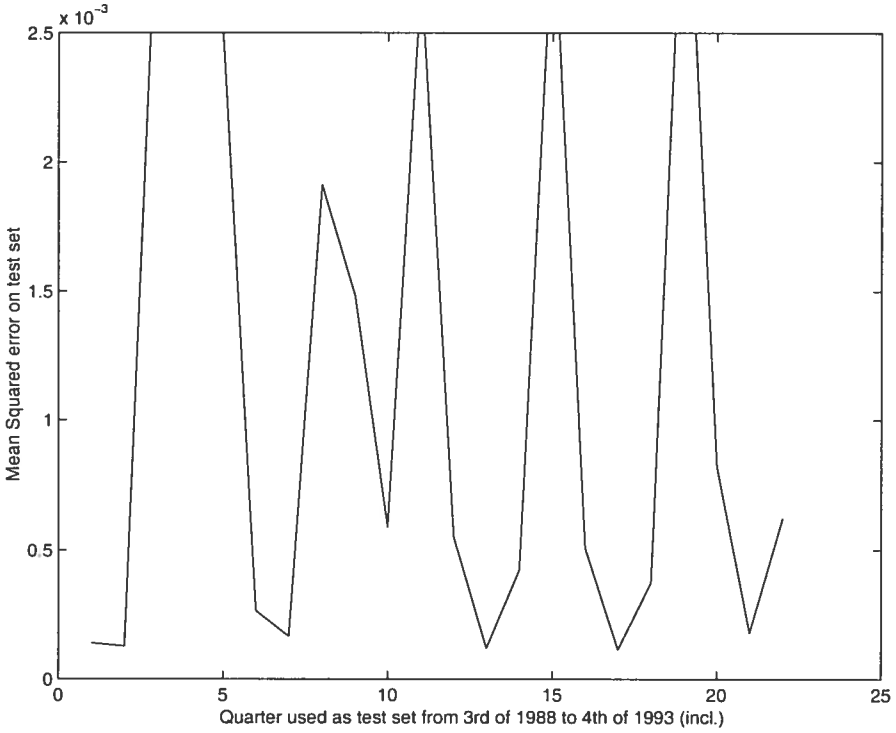

Figure 6.5: *Out-of-sample results from the third quarter of 1988 to the fourth of 1993 (incl.) for models with best validation results. Results for the unconstrained Black-Scholes similar network. Other unconstrained models exhibit similar swinging result patterns and levels of errors.*

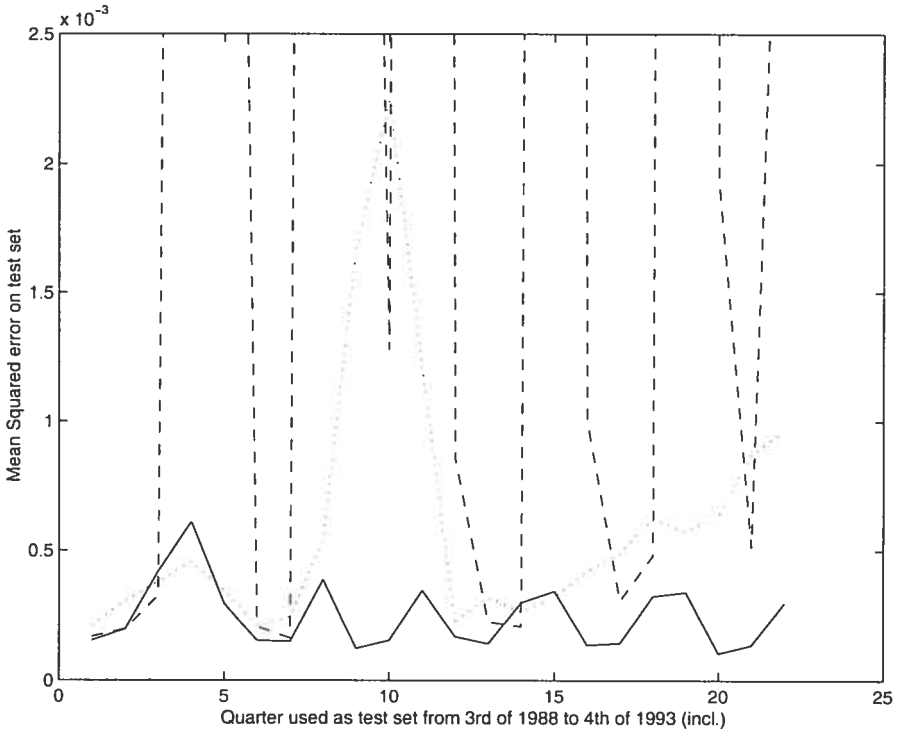

Figure 6.6: *Out-of-sample results from the third quarter of 1988 to the fourth of 1993 (incl.) for constrained models with best validation results. The fully constrained proposed architecture (solid) does best. The model with sums over dimensions obtains similar results. The regular neural network (dotted) is significantly worse. The constrained Black-Scholes model obtains very poor results (dashed).*

In another series of experiments, we tested the unconstrained multi-layered perceptron against the proposed constrained products of softplus convex architecture using data from years 1988 through 1993 incl. For each year, the first two quarters were used for training, the third quarter for model selection (validation) and the fourth quarter for testing. We trained neural networks for 50000 epochs and with a number of hidden units ranging from 1 through 10. In Table 6.5, we report training, validation and test results for the two chosen architectures. Model selection was performed using the validation set in order to choose the best number of hidden units, learning rate, learning rate decrease and weight decay. In all cases, except for 1988, the proposed architecture outperformed the multi-layered perceptron model. This might explain why the proposed architecture did not perform as well as other architectures on previous experiments using only data from 1988. Also note that the MSE obtained in 1989 is much higher. This is a possible explanation for the bad results obtained in tables 6.2 and 6.3 on the second test set. A hypothesis is that the process was undergoing nonstationarities that affected the forecasting performances. This shows that performance can vary by an order of magnitude from year to year and that forecasting in the presence of nonstationary processes is a difficult task.

6.7 Conclusions

Motivated by prior knowledge on the positivity of the derivatives of the function that gives the price of European options, we have introduced new classes of functions similar to multi-layer neural networks that have those properties. We have shown universal approximation properties for these classes. On simulation experiments, using artificial datasets, we have shown that these classes of functions lead to a reduction in the variance and the bias of the associated estimators. When applied in empirical tests of option pricing, we showed that the architecture from the proposed constrained classes usually generalizes better than a standard artificial neural network.

Simple Multi-Layered Perceptrons and Products of SoftPlus
Mean Squared Error Results on Call Option Pricing ($\times 10^{-5}$)

Year	Architecture	Units	Train	Valid	Test
1988	MLP	9	2.09	1.45	3.28
	Convex	9	3.86	2.70	5.23
1989	MLP	4	9.10	28.89	51.39
	Convex	2	9.31	23.96	48.22
1990	MLP	9	2.17	4.81	5.61
	Convex	9	1.58	4.18	5.39
1991	MLP	9	2.69	1.76	3.41
	Convex	8	2.62	1.25	2.74
1992	MLP	8	3.46	1.16	1.52
	Convex	8	3.27	1.28	1.27
1993	MLP	9	1.34	1.47	1.76
	Convex	10	0.68	0.54	0.65

Table 6.5: Comparison between a simple MLP architecture and the proposed architecture. Data from the first two quarters of each year was used as training set, data from the third quarter was used for validation and the fourth quarter was used for testing. We also report the number of units chosen by the model selection process.

6.8 Proof of the Universality Theorem for class ${}_{c,n}\hat{\mathcal{N}}_{++}$

In this section, we proove theorem 6.2.3. In order to help the reader through the formal mathematics, we first give an outline of the proof, i.e., a high-level informal overview of the proof. Then, in the first main part of the proof, we make use of two functions namely, the threshold $\theta(x) = I_{x \geq 0}$ and positive part $x_+ = max(0, x)$ functions. These two functions are part of the closure of the set ${}_{c,n}\hat{\mathcal{N}}_{++}$ since

$$\theta(x) = \lim_{t \to \infty} h(tx) \tag{6.23}$$

$$x_+ = \lim_{t \to \infty} \zeta(tx). \tag{6.24}$$

This extended class of functions shall be referred to as ${}_{c,n}\hat{\mathcal{N}}^{\infty}_{++}$. Indeed, as pointed out by Balazs Kegl, the proof, as it is stated in the first main part, only stands for the limit cases of class ${}_{c,n}\hat{\mathcal{N}}^{\infty}_{++}$ which are actually not part of class ${}_{c,n}\hat{\mathcal{N}}_{++}$. These cases call for the use of parameters of infinite value, making the proof without any practical bearing. For this reason, in the second main part, we broaden the theorem's application from ${}_{c,n}\hat{\mathcal{N}}^{\infty}_{++}$ to ${}_{c,n}\hat{\mathcal{N}}_{++}$, building upon the proof of the first main part.

6.8.1 Outline of the proof

The proof of the first main part works by construction: we start by setting the approximating function equal to a constant function. Then, we build a grid over the domain of interest and scan through it. At every point of the grid we add a term to the approximating function. This term is a function itself that has zero value in any point of the grid that has already been visited (or scanned). Thus, this term only affects the current point being visited and the points to be visited. The task is therefore to make sure the term being added is such that the approximating function matches the actual function at the point being visited. The grid structure and the convexity and monotonicity give guarantee for the values in the interior of the grid cells. The functions to be added are chosen from the set ${}_{c,n}\hat{\mathcal{N}}^{\infty}_{++}$ so that each of them individually respects the constraints on the derivatives. The bulk of the work in the proof is to show that, throughout the process, at each scanned point, we need to add a positive term to match the approximating function to the true function.

In the second part, we build upon the proof of the first part. The same constructive algorithm is used with the same increment values $\{\delta_h\}_{h=1}^{H}$. We simply consider sigmoidal and softplus functions that are greater or equal, in every point, than their limit counterparts, used in the first part. Consequently, there is a systematic overshoot of the function in every point. The main element of the second part is that the total overshoot is capped by $\epsilon/2$ in every grid point and ϵ in any other point of the domain. This can be done by setting the sigmoid and softplus parameter values appropriately. The approximating function is initialized to a value equal to $f(\vec{a}) - \epsilon/2$. The set of approximating functions is denoted $\{\tilde{f}_h\}_{h=1}^{H}$. The grid is made tighter than in the first part so that variations between neighbor points are bounded above by $\epsilon/2$.

6.8.2 Proof of the Universality Theorem for class ${}_{c,n}\hat{\mathcal{N}}_{++}$

Let D be the compact domain over which we wish to obtain an approximation error below ϵ in every point. Suppose the existence of an oracle allowing us to evaluate the function in a certain number of points. Let T be the smallest hyperrectangle encompassing D. Let us partition T in hypercubes with sides of length L so that the variation of the function between two neighboring points is bounded above by ϵ. For example, given s, an upper bound on the derivative of the function in any direction, setting $L \leq \epsilon/s$ would do the trick. Since the function to be approximated is Lipschitz, then its derivative is bounded and s does exist. The number of grid points is $N_1 + 1$ over the x_1 axis, $N_2 + 1$ over the x_2 axis, ..., $N_n + 1$ over the x_n axis. Thus, the number of points on the grid formed within T is $H = (N_1+1)\cdot(N_2+1)\cdot\ldots\cdot(N_n+1)$. We define grid points $\vec{a} = (a_1, a_2, \cdots, a_n)$ and $\vec{b} = (b_1, b_2, \cdots, b_n)$ as the innermost (closest to origin) and outermost corners of T, respectively, i.e., $T = \{x | a_i \leq x_i \leq b_i\}$. Figure 6.7 illustrates these values. The points of the grid are defined as:

$$\begin{aligned}
\vec{p}_1 &= a, \\
\vec{p}_2 &= (a_1, a_2, \ldots, a_n + L), \\
\vec{p}_{N_n+1} &= (a_1, a_2, \ldots, b_n), \\
\vec{p}_{N_n+2} &= (a_1, a_2, \ldots a_{n-1} + L, a_n), \\
\vec{p}_{N_n+3} &= (a_1, a_2, \ldots a_{n-1} + L, a_n + L), \ldots, \\
\vec{p}_{(N_n+1)(N_{n-1}+1)} &= (a_1, a_2, \ldots, a_{n-2}, b_{n-1}, b_n), \\
\vec{p}_{(N_n+1)(N_{n-1}+1)+1} &= (a_1, a_2, \ldots, a_{n-2} + L, a_{n-1}, a_n), \ldots, \\
\vec{p}_H &= b.
\end{aligned} \tag{6.25}$$

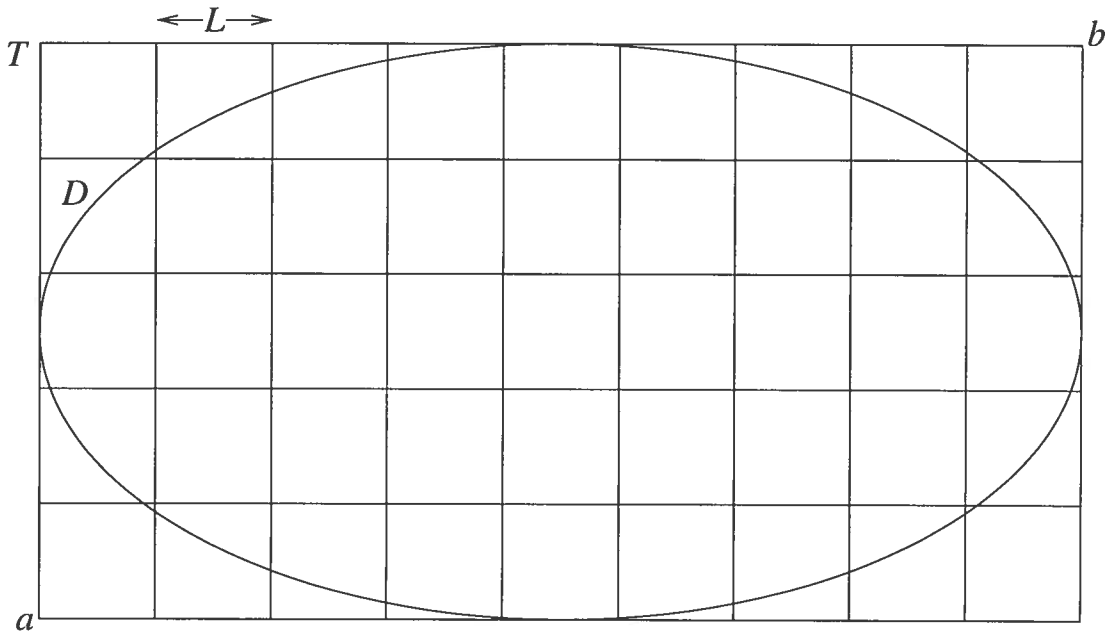

Figure 6.7: *Two dimensional illustration of the proof of universality: The ellipse D corresponds to the domain of observation over which we wish to obtain a universal approximator. The rectangle T encompasses D and is partitioned in squares of length L. Points a and b are the innermost (closest to origin) and outermost corners of T, respectively.*

We start with an approximating function $\hat{f}_0 = f(\vec{a})$, i.e., the function $\hat{f}_0$ is initially set to a constant value equal to $f(\vec{a})$ over the entire domain. Note that, for the remainder of the proof, notations $\hat{f}_h$, f_h, g_h, without any argument, refer to the functions themselves. When an argument is present, such as in $f_h(\vec{p})$, we refer to the value of the function f_h evaluated at point $\vec{p}$.

After setting $\hat{f}_0$ to its initial value, we scan the grid in an orderly manner, according to the order defined in equation 6.25. At each point along the grid, we add a term (g_h, a function) to the current approximating function so that it becomes exact at point $\{\vec{p}_h\}$:

$$\begin{aligned} \hat{f}_h &= g_h + \hat{f}_{h-1} \\ &= \sum_{k=0}^{h} g_k \end{aligned} \tag{6.26}$$

where we have set $g_0 = \hat{f}_0$.

The functions $\hat{f}_h$, g_h and $\hat{f}_{h-1}$ are defined over the whole domain and the increment function g_h must be such that at point $\vec{p}_h$, we have $\hat{f}_h(\vec{p}_h) = f(\vec{p}_h)$. We compute the constant term δ_h as the difference between the value of the function evaluated at point $\vec{p}_h$, $f(\vec{p}_h)$, and the value of the currently accumulated approximating function at the same point $\hat{f}_{h-1}(\vec{p}_h)$:

$$\delta_h = f(\vec{p}_h) - \hat{f}_{h-1}(\vec{p}_h). \tag{6.27}$$

Now, the function g_h must not affect the value of the approximating function at grid points that have already been visited. According to our sequencing of the grid points, this corresponds to having $g_h(\vec{p}_k) = 0$ for $0 < k < h$. Enforcing this constraint ensures that $\hat{f}_h(\vec{p}_k) = \hat{f}_k(\vec{p}_k) = f(\vec{p}_k)$, $0 < k < h$. We define

$$\beta(\vec{y}, \vec{z}) = \prod_{j=1}^{c} (y(j) - z(j) + L)_+ \cdot \prod_{j=c+1}^{n} \theta(y(j) - z(j)) \tag{6.28}$$

where $y(j)$ is the j^{th} coordinate of $\vec{y}$ and similarly for $\vec{z}$. We have assumed, without loss of generality, that the convex dimensions are the first c ones. One can readily verify that $\beta(\vec{p}_k, \vec{p}_h) = 0$ for $0 < k < h$. We can now define the incremental term:

$$g_h(\vec{p}) = \delta_h \beta(\vec{p}, \vec{p}_h) \tag{6.29}$$

so that after all grid points have been visited, our final approximation is

$$\hat{f}_H(\vec{p}) = \sum_{k=0}^{H} g_k(\vec{p}) \tag{6.30}$$

with $f(\vec{p}) = \hat{f}_H(\vec{p})$ for all grid points.

So far, we have devised a way to approximate the target function as a sum of terms from the set ${}_{c,n}\hat{\mathcal{N}}_{++}$. We know our approximation to be exact in every point of a grid and that the grid is tight enough so that the approximation error is bounded above by ϵ anywhere else: take any two neighbor points $\vec{q}_1$ and $\vec{q}_2$ where, according to the algorithm, the approximating function matches exactly the target function, i.e., $\hat{f}_H(q_1) = f(q_1)$ and $\hat{f}_H(q_2) = f(q_2)$. These points are at a distance L along a certain axis and we know the slope of the function is bounded by s on the entire domain of interest. Since we have set $L \leq \epsilon/s$, then $|f(q_1) - f(q_2)| \leq Ls \leq \epsilon$. Given the set of constraints in equation 6.8, the target function must be monotonic between point q_1 and q_2. Also, the approximating function is composed of strictly monotonic terms. Without loss of generality, assume $f(q_1) \leq f(q_2)$. Take any point $\tilde{q} = tq_1 + (1-t)q_2$, $t \in [0,1]$ between q_1 and q_2. Again, without loss of generality, assume $\hat{f}_H(\tilde{q}) < f(\tilde{q})$. Then $f(\tilde{q}) - \hat{f}_H(\tilde{q}) < f(q_2) - \hat{f}(q_1) = f(q_2) - f(q_1) \leq \epsilon$.

The target function respects a set of constraints on its derivatives as expressed in equation 6.8. The terms of our final approximate function $\hat{f}_H$ also individually respect the constraints. One essential question remains: does $\hat{f}_H$, the sum of all these terms, also respect these constraints ? In order to ensure this, we need to show that $\delta_h \geq 0 \; \forall h$.

Let $p_k(j)$ be the value of the j^{th} coordinate of point $\vec{p}_k$. Also, let $p_k(j) = a(j) + i_k(j)L$. In other words, $\vec{p}_k = \vec{a} + L \cdot (i_k(1), i_k(2), \ldots, i_k(n))$ and this defines the bijective relationship between k and the $\{i_k(j)\}_{j=1}^{n}$ values. Note that for any pair of points $(\vec{y}, \vec{z})$, we have $y(j) - z(j) = L(i_y(j) - i_z(j))$. Let us first express the target function in terms of the δ values.

First, according to equation 6.30, we have:

$$\begin{aligned} f(\vec{p}_h) &= \hat{f}_H(\vec{p}_h) \\ &= \sum_{k=0}^{H} g_k(\vec{p}_h) \end{aligned} \tag{6.31}$$

Looking at equations 6.46 and 6.29, we see that $g_h(\vec{p})$ is equal to zero if $p(j) > p_h(j)$ for any j. By summing on non-zero values and replacing the g_h terms according to these two equations, we obtain:

$$\begin{aligned} f(\vec{p}_h) &= \sum_{\{k:p_k(j)\leq p_h(j)\,\forall j\}} g_k(\vec{p}_h) \\ &= \sum_{\{k:p_k(j)\leq p_h(j)\,\forall j\}} \delta_k \prod_{j=1}^{c}(p_h(j) - p_k(j) + L)_+ \cdot \prod_{j=c+1}^{n} \theta(p_h(j) - p_k(j)) \end{aligned} \tag{6.32}$$

Since $p_h(j) > p_k(j)$ for each j of each of the remaining terms in the summation, then $\theta(p_h(j) - p_k(j)) = 1$ for every term:

$$\begin{aligned} f(\vec{p}_h) &= \sum_{\{k:p_k(j)\leq p_h(j)\,\forall j\}} \delta_k \prod_{j=1}^{c}(i_h(j) - i_k(j) + 1)L \cdot \prod_{j=c+1}^{n} 1 \\ &= \sum_{i_k(1)=1}^{i_h(1)} \sum_{i_k(2)=1}^{i_h(2)} \cdots \sum_{i_k(n)=1}^{i_h(n)} \delta_k L^c \prod_{j=1}^{c}(i_h(j) - i_k(j) + 1) \end{aligned} \tag{6.33}$$

where the last equation is obtained by making use of the definition of the set of $i_k(j)$ values.

Then, we define the finite difference of the function along the l^{th} axis as

$$\Delta_l f(\vec{p}_h) = f(\vec{p}_h) - f(\vec{p}_{k:i_k(l)=i_h(l)-1}) \tag{6.34}$$

so that k is a neighbor of h on the hypergrid of points within T. All coordinates of k and h are the same except along the l^{th} axis where $i_k(l) = i_h(l) - 1$. Using equations 6.34 and 6.33 we get:

$$\begin{aligned} \Delta_l f(\vec{p}_h) &= \sum_{i_k(1)=1}^{i_h(1)} \cdots \sum_{i_k(l)=1}^{i_h(l)} \cdots \sum_{i_k(n)=1}^{i_h(n)} \delta_k L^c \prod_{j=1}^{c}(i_h(j) - i_k(j) + 1) \\ &- \sum_{i_k(1)=1}^{i_h(1)} \cdots \sum_{i_k(l)=1}^{i_h(l)-1} \cdots \sum_{i_k(n)=1}^{i_h(n)} \delta_k L^c \prod_{j=1}^{c}(i_h(j) - i_k(j) + 1) \end{aligned} \tag{6.35}$$

We then reorder the summations so that the l^{th} axis comes last. We also factor out the terms that are independent of l. Let us first consider the case

where $l \leq c$, i.e., the l^{th} dimension bears the convexity property:

$$\Delta_l f(\vec{p}_h) = \sum_{i_k(1)=1}^{i_h(1)} \cdots \sum_{i_k(n)=1}^{i_h(n)} L^c \prod_{j=1, j\neq l}^{c} (i_h(j) - i_k(j) + 1) \cdot \alpha(l)$$

where,

$$\begin{aligned} \alpha(l) &= \sum_{i_k(l)=1}^{i_h(l)} \delta_k(i_h(l) - i_k(l) + 1) - \sum_{i_k(l)=1}^{i_h(l)-1} \delta_k(i_h(l) - 1 - i_k(l) + 1) \\ &= \sum_{i_k(l)=1}^{i_h(l)} \delta_k \end{aligned} \tag{6.36}$$

so that $\Delta_l f(\vec{p}_h)$ still bears a dependency on its l^{th} dimension. We therefore differentiate once more along the l^{th} axis:

$$\begin{aligned} \Delta_l^2 f(\vec{p}_h) &= \sum_{i_k(1)=1}^{i_h(1)} \cdots \sum_{i_k(n)=1}^{i_h(n)} L^c \prod_{j=1, j\neq l}^{c} (i_h(j) - i_k(j) + 1) \cdot \left(\sum_{i_k(l)=1}^{i_h(l)} \delta_k - \sum_{i_k(l)=1}^{i_h(l)-1} \delta_k \right) \\ &= \sum_{i_k(1)=1}^{i_h(1)} \cdots \sum_{i_k(n)=1}^{i_h(n)} \delta_{k:i_k(l)=i_h(l)} L^c \prod_{j=1, j\neq l}^{c} (i_h(j) - i_k(j) + 1) \end{aligned} \tag{6.37}$$

so that we remove any dependency on the l^{th} dimension. Now in case $l > c$, our task is simpler since:

$$\begin{aligned} \alpha(l) &= \sum_{i_k(l)=1}^{i_h(l)} \delta_k - \sum_{i_k(l)=1}^{i_h(l)-1} \delta_k \\ &= \delta_{k:i_k(l)=i_h(l)} \end{aligned}$$

and we obtain:

$$\Delta_l f(\vec{p}_h) = \sum_{i_k(1)=1}^{i_h(1)} \cdots \sum_{i_k(n)=1}^{i_h(n)} \delta_{k:i_k(l)=i_h(l)} L^c \prod_{j=1, j\neq l}^{c} (i_h(j) - i_k(j) + 1) \tag{6.38}$$

Note the similarity of equations 6.37 and 6.38. Both remove dependency along the dimension of differentiation. They address the cases where $l \leq c$ and $l > c$, respectively. This procedure can be applied recursively over all dimensions so that in the end,

$$\begin{aligned} \Delta_1^2 \dots \Delta_c^2 \Delta_{c+1} \dots \Delta_n f(\vec{p}_h) &= L^c \delta_{k:i_k(1)=i_h(1)\dots i_k(n)=i_h(n)} \\ &= L^c \delta_h \end{aligned} \tag{6.39}$$

and we have finally isolated the value of δ_h as a function of L and a finite difference of order $n + c$ of the function. The value of δ_h is non negative iff this finite difference value also is.

Now, by definition of the integral operator,

$$\Delta f = \frac{f(b)-f(a)}{b-a} = \frac{1}{b-a} \int_a^b f' \, dx \tag{6.40}$$

so that if $f' \geq 0$ over the range $[a, b]$, then consequentely, $\Delta f \geq 0$. Applying this to our case, we set

$$\frac{\partial^{n+c} f(p_h)}{\partial x_1^2 \partial x_2^2 \dots \partial x_c^2 \partial x_{c+1} \dots \partial x_n} \geq 0 \tag{6.41}$$

over the hyperrectangle T so that

$$\Delta_1^2 \dots \Delta_c^2 \Delta_{c+1} \dots \Delta_n f(\vec{p}_h) \geq 0 \tag{6.42}$$

over T as well and consequently, $\delta_h \geq 0 \, \forall h$. □

6.8.3 Illustration of the constructive algorithm

In order give the reader a better intuition as to how we were able to isolate the δ_h factor in equation 6.39, we apply the finite difference method to ${}_{1,2}\hat{\mathcal{N}}_{++}$, the set of functions that include call price functions, i.e., positive convex w.r.t. the first variable and monotone increasing w.r.t. both variables. Figure 24 illustrates the two dimensional setting of our example with the points of the grid labelled in the order in which they are scanned according the constructive procedure. We will show how to isolate δ_6.

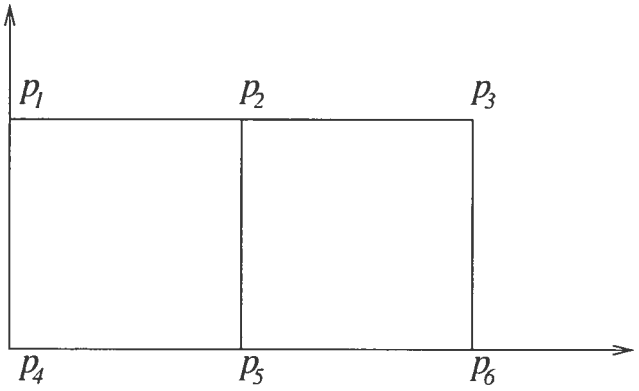

Figure 6.8: *Illustration in two dimensions of the constructive proof. The grid is scanned along the abscissa axis, then along the ordinates axis. The points are labelled accordingly from 1 to 6. The function is known to be convex w.r.t. to the first variable (abscissa) and monotone increasing w.r.t. both variables.*

For the set ${}_{1,2}\hat{\mathcal{N}}_{++}$, we have,

$$f(\vec{p}_h) = \sum_{k=1}^{H} \delta_k \cdot (p_h(1) - p_k(1) + L)_+ \cdot \theta(p_h(2) - p_k(2)) \qquad (6.43)$$

Applying this to the six grid points of Figure 24, we obtain $f(\vec{p}_1) = L\delta_1$, $f(\vec{p}_2)=L(\delta_1 + \delta_2)$, $f(\vec{p}_3) = L(2\delta_1 + \delta_3)$, $f(\vec{p}_4) = L(2\delta_1 + 2\delta_2 + \delta_3 + \delta_4)$, $f(\vec{p}_5) = L(3\delta_1 + 2\delta_3 + \delta_5)$, $f(\vec{p}_6) = L(3\delta_1 + 3\delta_2 + 2\delta_3 + 2\delta_4 + \delta_5 + \delta_6)$.

Differentiating w.r.t. the second variable, then the first, we have:

$$\begin{aligned}
\Delta_2 f(\vec{p}_6) &= f(\vec{p}_6) - f(\vec{p}_5) \\
\Delta_1\Delta_2 f(\vec{p}_6) &= (f(\vec{p}_6) - f(\vec{p}_4)) - (f(\vec{p}_5) - f(\vec{p}_3)) \\
\Delta_1\Delta_2 f(\vec{p}_6) &= (f(\vec{p}_6) - f(\vec{p}_4)) - (f(\vec{p}_4) - f(\vec{p}_2)) \\
&- (f(\vec{p}_5) - f(\vec{p}_3)) + (f(\vec{p}_3) - f(\vec{p}_1)) \\
&= \delta_6
\end{aligned}$$

This procedure can be repeated for any point on the grid. The conclusion associated with this result is that the third finite difference of the function must be positive in order for δ_6 to be positive as well. As stated above, enforcing the corresponding derivative is a slightly stronger condition which is respected by all element functions of ${}_{1,2}\hat{\mathcal{N}}_{++}$.

For points close to the boundary, it is simpler to isolate the δ_h value as fewer finite difference values need to be computed. The constraint for the associated δ_h values is therefore set on derivatives of lower orders which are still respected by all elements of ${}_{1,2}\hat{\mathcal{N}}_{++}$.

6.8.4 Proof of the Universality Theorem for class ${}_{c,n}\hat{\mathcal{N}}_{++}$

We start with an approximating function $\tilde{f}_0 = f(\vec{a}) - \epsilon/2$, i.e., the function $\tilde{f}_0$ is initially set to a constant value equal to $f(\vec{a}) - \epsilon/2$ over the entire domain.

After setting $\tilde{f}_0$ to its initial value, we scan the grid in an orderly manner, according to the definition of the set of points $\{\vec{p}_h\}$. At each point $\vec{p}_h$ along the grid, we add a term $\tilde{g}_h$ (a function) to the current approximating function

$\tilde{f}_{h-1}$:

$$\begin{aligned}\tilde{f}_h &= \tilde{g}_h + \tilde{f}_{h-1} \\ &= \sum_{k=1}^{h} \tilde{g}_k \qquad (6.44) \\ &= \sum_{k=1}^{h} \delta_k \tilde{\beta}_{\vec{p}_k} \qquad (6.45)\end{aligned}$$

where the δ_k are equal to the ones found in the previous proof, and we define the set of $\tilde{\beta}_{\vec{p}_k}$ functions as a product of sigmoid and softplus functions, one for each input dimension:

$$\tilde{\beta}_{\vec{p}_h}(\vec{y}) = \prod_{j=1}^{n} \tilde{\beta}_{\vec{p}_h,j}(\vec{y}). \qquad (6.46)$$

where $p_h(j)$ is the ^{e}j coordinate of vector $\vec{p}_h$. For each of the convex coordinates, we set:

$$\tilde{\beta}_{\vec{p}_h,j}(\vec{y}) = \frac{1}{\alpha}\zeta(\alpha \cdot (y(j) - p_h(j))). \qquad (6.47)$$

Now, note that the maximum of the difference between the softplus function as defined in equation 6.47 and the positive part function $(y(j) - p_h(j))_+$ is attained for $y(j) = p_h(j)$ with value $\kappa = \ln 2/\alpha$. Thus, in order to cap the overshoot resulting from the approximation along the convex dimensions, we simply need to set κ (α) to a small (large) enough value which we shall define later. Let us now turn to the non-convex dimensions where set:

$$\tilde{\beta}_{\vec{p}_h,j}(\vec{y}) = (1+\kappa)\, h(\alpha \cdot y(j) + \gamma) \qquad (6.48)$$

and add two constraints:

$$h(\alpha(p_h(j) - L) + \gamma) = \frac{\kappa}{1+\kappa} \qquad (6.49)$$

$$h(\alpha \cdot p_h(j) + \gamma) = \frac{1}{1+\kappa}. \qquad (6.50)$$

Solving for α and γ, we obtain:

$$\alpha = -\frac{2}{L}\ln\kappa \qquad (6.51)$$

$$\gamma = \left(\frac{2p_h(j)}{L} - 1\right)\ln\kappa. \qquad (6.52)$$

Thus, for values of $y(j)$ such that $p_h(j)-L < y(j) < p_h(j)$, we have a maximal overshoot of 1. For other values of $y(j)$, the overshoot is capped by κ. In particular, the overshoot is bounded above by κ for $y(j) = p_h(j) + nL\ n \in \mathbb{Z}$ and zero for $y(j) = p_h(j)$. We now compare incremental terms. Our goal is to cap the difference between $\tilde{g}_h$ and $\hat{g}_h$ at grid points:

$$\tilde{g}_h - \hat{g}_h = \delta_h(\tilde{\beta}_{p_h} - \hat{\beta}_{p_h}). \tag{6.53}$$

At grid points, $\hat{\beta}_{p_h}$ is equal to

$$\hat{\beta}_{p_h} = L^c \prod_{j=1}^{n} m_j \tag{6.54}$$

where $m_j \in \mathbb{N}$ and is equal to 0 or 1 along the non-convex dimensions and equal to a non-negative integer along the convex dimensions. Similarly, we have

$$\tilde{\beta}_{p_h} \leq L^c \prod_{j=1}^{n} (m_j + \kappa) \tag{6.55}$$

First consider the case where $d > 0$ of the m_j's are equal to zero. Then,

$$\begin{aligned}
\tilde{g}_h - \hat{g}_h &\leq \delta_h L^c (\prod_{j=1}^{n} (m_j + \kappa) - \prod_{j=1}^{n} m_j) \\
&\leq \delta_h L^c \kappa^d \prod_{j=1, m_j \neq 0}^{n} m_j (1 + \kappa) \\
&\leq \delta_h L^c H \kappa^d (1 + \kappa)^n \\
&\leq \epsilon L^c H \kappa (1 + \kappa)^n.
\end{aligned}$$

And, assuming $\kappa < 1$, we obtain $\tilde{g}_h - \hat{g}_h < \epsilon/2H$ if

$$\kappa < \frac{1}{2^{n+1} L^c H^2} \tag{6.56}$$

If none of the m_j's is equal to zero, then similarly,

$$\begin{aligned}
\tilde{g}_h - \hat{g}_h &\leq \delta_h L^c (\prod_{j=1}^{n}(m_j + \kappa) - \prod_{j=1}^{n} m_j) \\
&\leq \delta_h L^c (\prod_{j=1}^{n} m_j(1+\kappa) - \prod_{j=1}^{n} m_j) \\
&\leq \delta_h L^c H((1+\kappa)^n - 1) \\
&\leq \epsilon L^c H((1+\kappa)^n - 1).
\end{aligned}$$

In that case, we have that $\tilde{g}_h - \hat{g}_h < \epsilon/2H$ if

$$\kappa < \left(1 + 1/2H^2L^c\right)^{1/n} - 1 \tag{6.57}$$

Which of equation 6.56 or equation 6.56 gives the tighest bound depends on problem specific parameter values. Nonetheless, we have shown that the difference between the approximating function obtained without the use of the threshold and positive part functions can be capped by $\epsilon/2H$ at grid points.

Now, consider point $\vec{y}$ within the grid between grid points $\vec{p}_1$ and $\vec{p}_2$, with $g_h(\vec{p}_2) > g_h(\vec{p}_1)$:

$$\begin{aligned}
\tilde{g}_h(\vec{y}) - \hat{g}_h(\vec{y}) &\leq \tilde{g}_h(\vec{p}_2) - \hat{g}_h(\vec{p}_1) \\
&\leq \tilde{g}_h(\vec{p}_2) - \hat{g}_h(\vec{p}_2) + \hat{g}_h(\vec{p}_2) - \hat{g}_h(\vec{p}_1) \\
&\leq \epsilon/H
\end{aligned}$$

Looking at the approximating functions, we have

$$\begin{aligned}
\tilde{f}_H(\vec{y}) - \hat{f}_H(\vec{y}) &= \sum_{k=1}^{H} \tilde{g}_k(\vec{y}) - \sum_{k=1}^{H} \hat{g}_k(\vec{y}) \\
&\leq \epsilon
\end{aligned}$$

Now, we have assumed the grid to be tight enough so that

$$f(\vec{y}) - \epsilon/2 < \hat{f}_H(\vec{y}) < f(\vec{y}) + \epsilon/2. \tag{6.58}$$

Taking account that we have set $\tilde{f}_0 = f(\vec{a}) - \epsilon/2$

$$f(\vec{y}) - \epsilon < \hat{f}_H(\vec{y}) - \epsilon/2 < f(\vec{y}). \tag{6.59}$$

Finally, adding the overshoot leads to the desired result:

$$f(\vec{y}) - \epsilon < \quad \tilde{f}_H(\vec{y}) \quad < f(\vec{y}) + \epsilon. \tag{6.60}$$

□

Corollary 6.8.1 *Within the set of positive Lipschitz functions from* $\mathbb{R}$ *to* $\mathbb{R}$ *whose first and second derivatives are non-negative, the class* ${}_{1,1}\hat{\mathcal{N}}_{++}$ *is a universal approximator.*

Chapitre 7

Synthèse

À la section 6.3, nous avons réalisé une série d'expériences avec des données artificielles dans le but de mesurer l'amélioration causée par l'utilisation de l'architecture proposée. L'analyse portait sur l'évolution du biais et de la variance en fonction du niveau de bruit induit dans le données et la taille de l'ensemble d'entraînement. À une exception près, tous les résultats montrent une réduction simultanée du biais et de la variance lorsque l'architecture proposée est utilisée. La réduction de la variance correspond à l'imposition de contraintes appropriées et donc la réduction de l'espace des fonctions considérées. Nous conjecturons que la réduction du biais est due à l'atteinte d'un nouveau point d'équilibre dans la résolution du dilemme biais-variance. Les améliorations sont généralement plus importantes lorsque la taille de l'ensemble d'entraînement est plus faible.

Les tableaux 6.2 et 6.3 montrent que les erreurs d'entraînement, de validation et de test (test 1) sont inférieures pour les architectures non contraintes. Toutefois, les performances de ces architectures se dégradent substantiellement lorsque l'on performe un second test (test 2), avec les données du trimestre suivant. Si l'on regarde sur une période de quelques années, on constate (figures 6.5 et 6.6) que la performance des architectures contraintes est supérieure. Nous conjecturons que les architectures proposées (contraintes) détectent des relations fondamentales, plus robustes aux non-stationarités propres aux données de séries chronologiques, ce qui explique cette évolution des résultats dans le temps.

Les résultats du tableau 6.5 montrent que, à l'exception de l'année 1988, l'architecture contrainte proposée performe mieux que l'architecture non contrainte. Le fait que l'architecture proposée ait moins bien performé que les

autres architectures peut donc s'expliquer par un choix malhanceux. Notons aussi que les erreurs obtenues en 1989 sont largement supérieures aux autres années, ce qui peut expliquer les mauvais résultats des tableaux 6.2 et 6.3 sur le second test.

Nous basant sur la connaissance *a priori* de propriétés sur la fonction du prix des options européennes et de certaines de ses dérivées, nous avons introduit une nouvelle architecture de réseaux de neurones. Nous avons montré que cette nouvelle classe de fonctions est un approximateur universel. Sur des données artificielles, nous avons obtenu une réduction simultanée du biais et de la variance. Sur des données d'options, nous avons montré que l'architecture proposée permet généralement d'obtenir de meilleurs résultats que les réseaux de neurones standards.

Troisième partie

Troisième article

Chapitre 8

Introduction

Ce chapitre de livre porte sur l'application des réseaux de neurones à la tarification en assurance automobile, un des principaux problèmes mathématiques auxquels se trouvent confrontés les actuaires. On doit tout d'abord évaluer l'espérance mathématique des réclamations pour une police d'assurance. Cette valeur représente la prime pure à laquelle s'ajouteront les frais d'administration de la police pour obtenir la prime brute que l'assuré paiera. Le chapitre se concentre sur l'utilisation d'algorithmes d'apprentissage pour l'évaluation de la prime pure. Cette valeur dépend de plusieurs facteurs explicatifs, généralement une trentaine.

Étant donné le nombre important de ces facteurs explicatifs, le problème de l'évaluation de la prime pure peut être considéré comme étant frappé de la malédiction de la dimensionalité (6), c.-à-d. qu'il est impossible de tenir compte de l'effet de chacune des combinaisons de valeurs différentes des facteurs explicatifs. À l'opposé, l'hypothèse d'indépendance entre les variables peut causer d'importantes dégradations de la performance des modèles prédictifs (4). Les réseaux de neurones, eux, sont reconnus pour leur capacité à détecter les combinaisons de valeurs les plus probantes et les représenter avec un nombre limité de paramètres (75). Cette propriété particulière nous a conduits vers l'avenue de recherche que nous présentons au chapitre suivant.

La présence de grandes réclamations et donc d'une distribution à queue longue et épaisse nous oblige à utiliser des méthodes d'estimation robustes. Les approches traditionnelles (51; 52; 53; 54; 55; 57; 58; 43; 74) permettent de tenir compte de distributions bruitées mais symmétriques, ce qui n'est pas le cas des montants de réclamations en assurance automobile. Des percées récentes (82) ont été réalisées dans le but de solutionner ce problème particu-

lier et nous proposons une architecture particulière à cette fin qui est basée sur une mixture d'experts (60).

Les algorithmes proposés sont comparés à divers algorithmes existants : régressions linéaires, modèles linéaires généralisés (71; 67; 15; 49; 69), arbres de décisions (61; 14), réseaux de neurones standards et machines à vecteurs de support (84; 78; 22; 77).

Le chapitre débute par une mise en relief des percées importantes de l'inférence statistique que nous considérons classique et essentiellement due aux travaux de Fisher (34; 35; 36; 37; 2) et celles qui sont à la base du développement plus tardif de l'apprentissage statistique (42; 17; 63). Par le biais d'un exemple (41; 70), nous montrons que les deux approches mènent à des tests d'hypothèse différents et donc nécessairement des conclusions potentiellement différentes lorsque l'on considère l'ajout d'un paramètre. Dans un second exemple, les deux approches mènent à des estimateurs différents puisque l'apprentissage statistique fait usage de la méthode de régularisation (83) afin de résoudre le dilemme biais-variance (39), ce qui nous mène vers des modèles à capacité (84; 45) inférieure.

Le chapitre se poursuit avec une section dans laquelle les objectifs mathématiques sont décrits, suivie d'une section portant sur la méthodologie de sélection de modèle. Puis, les divers algorithmes sont décrits, les résultats expérimentaux sont présentés et une conclusion termine le document.

Les résultats de cette recherche ont donné lieu à un article de conférence (19), un article de revue (32), une présentation orale lors du « Seminar on Ratemaking 2003 » de la « Casualty Actuarial Society » et le chapitre de livre qui suit (31). L'auteur fut le principal rédacteur de cet ouvrage, fut responsable de la soumission et de l'apport des corrections nécessaires. L'auteur a aussi réalisé toutes les expériences de la section 9.7 et effectué la présentation en conférence, mentionnée plus haut.

Chapter 9

Statistical Learning Algorithms Applied to Automobile Insurance Ratemaking

The chapter will start from a description of the fundamentals of statistical learning algorithms and highlight how its basic tenets and methodologies differ from those generally followed by actuaries and econometricians. The main premise is that reality is too complex to be captured with a single unifying model, although some aspects may be well approximated by models. Therefore the statistical learning approach does not presume that reality is perfectly captured by a model, or at least tries to minimize the assumptions about the true generating distribution of the data. The approach is empirical: good models will be distinguished from poor models by comparing their predictive power and explanatory power on new data. At this point it is interesting to consider that choosing among models may be guided by two different objectives, which sometimes lead to different answers: an operational objective (which model will make the best decisions/predictions on new data), or a "modeling" objective (which model better describes the true underlying nature of the data). We will show an example in which the two approaches lead to different statistical tests and the operational approach makes more conservative decisions (chooses simpler models). Another example of the difference between the two approaches is illustrated by the case of ridge regression: there is a regularized (biased) regression that brings better out-of-sample expected predictions than the maximum likelihood (unbiased) estimator. This example will be used to illustrate the famous bias-variance

dilemma that is so pervasive in statistical learning algorithms.

The above discussion and introduction to the principles of statistical learning will naturally bring up the issue of methodology. We will describe and justify the main methodological tools of the statistical learning approach for selecting and comparing models, either based on theoretical bounds or on resampling techniques (such as the cross-validation and bootstrap techniques). A special section on the particular (and rarely discussed) issue of non-stationary data will explain how the above resampling methods can be generalized to data whose distribution varies over time, which is the case with insurance data. In order to evaluate and compare models, one needs to build statistical tools to evaluate the uncertainty in the measurements of out-of-sample performance (due to finite data and non-stationarity).

We applied the principles and methodology described above in a research contract we recently conducted for a large North American automobile insurer. This study was the most exhaustive ever undertaken by this particular insurer and lasted over an entire year. We analyzed the discriminative power of each variable used for ratemaking. We analyzed the performance of several statistical learning algorithms within five broad categories: Linear Regressions, GLMs, Decision Trees, Neural Networks and Support Vector Machines. We present the main results of this study.

We qualitatively compare models and show how Neural Networks can represent high order nonlinear dependencies with a small number of parameters, each of which is estimated on a large proportion of the data thus, has low variance. We thoroughly explain the purpose of the nonlinear sigmoidal transforms which are at the very heart of Neural Networks' performances. The main numerical result is a statistically significant reduction in the out-of-sample mean-squared error using the Neural Network model.

In some provinces and states, better risk discrimination, if not directly implemented because of market share concerns or legislative constraints, can also be used for the purpose of choosing the risks to be sent to "risk-sharing pools". According to these plans, insurers choose a portion of their book of business which they ceed to the pool. Losses (seldom gains) are shared by participants and/or insurers doing business in the province or state of the plan. Since the selection of risks to be sent to the pool bears no effect on market share (the insured is unaware of the process) and legislation is generally looser than that of ratemaking, highly discriminative statistical learning algorithms such as Neural Networks can be very profitably used to identify those most underpriced risks that should be ceeded to the pool. We

compare Generalized Linear Models to our Neural Network based model with respect to their risk-sharing pool performance.

9.1 Introduction

Ratemaking is one of the main mathematical problems faced by actuaries. They must first estimate how much each insurance contract is expected to cost. This conditional expected claim amount is called the *pure premium* and it is the basis of the *gross premium* charged to the insured. This expected value is conditioned on information available about the insured and about the contract, which we call the *input profile*.

Automobile insurance ratemaking is a complex task for many reasons. First of all, many factors are relevant. Taking account of each of them individually, i.e., making independence assumptions, can be hurtful (4). Taking account of all interactions is intractable and is sometimes referred to as the *curse of dimensionality* (6). In practice, actuarial judgment is used to discover the most relevant of these interactions and feed them explicitly to the model. Neural networks, on the other hand, are well-known for their ability to represent high order nonlinear interactions with a small number of parameters, i.e., they can automatically detect those most relevant interactions between variables (75). We explain how and why in section 9.5.

A second difficulty comes from the distribution of claims: asymmetric with fat tails with a large majority of zeros and a few unreliable and very large values, i.e., an asymmetric heavy tail extending out toward high positive values. Modeling data with such a distribution is essentially difficult because *outliers*, which are sampled from the tail of the distribution, have a strong influence on parameter estimation. When the distribution is symmetric around the mean, the problems caused by outliers can be reduced using *robust* estimation techniques (58; 43; 74) which basically intend to ignore or down-weight outliers. Note that these techniques do not work for an asymmetric distribution: most outliers are on the same side of the mean, so down-weighting them introduces a strong bias on its estimation: the conditional expectation would be systematically underestimated. Recent developments for dealing with asymmetric heavy-tail distributions have been made (82).

The third difficulty is due to the non-stationary nature of the relationship between explanatory variables and the expected claim amount. This has an important effect on the methodology to use, in particular with respect to the

task of *model selection.* We describe our methodology in section 9.4.

Fourth, from year to year, the general level of claims may fluctuate heavily, in particular in states and provinces where winter plays an important role in the frequency and severity of accidents. The growth of the economy and the price of gas can also affect these figures.

Fifth, one needs sufficient computational power to develop models: we had access to a large database of $\approx 8 \times 10^6$ records, and the training effort and numerical stability of some algorithms can be burdensome for such a large number of training examples.

Sixth, the data may be of poor quality. In particular, there may be missing fields for many records. An actuary could systematically discard incomplete records but this leads to loss of information. Also, this strategy could induce a bias if the absence of a data is not random but rather correlated to some particular feature which affects the level of risk. Alternatively one could choose among known techniques for dealing with missing values (26; 40; 10).

Seventh, once the pure premiums have been established the actuary must properly allocate expenses and a reserve for profit among the different contracts in order to obtain the gross premium level that will be charged to the insureds. Finally, an actuary must account for competitive concerns: his company's strategic goals, other insurers' rate changes, projected renewal rates and market elasticity.

In this chapter, we address the task of setting an appropriate pure premium level for each contract, i.e., difficulties one through four as described above. Our goal is to compare different models with respect to their performance in that regard, i.e., how well they are able to forecast the claim level associated to each contract. We chose several models within five broad categories: Linear Regressions, Generalized Linear Models (67), Decision Trees (61), Neural Networks and Support Vector Machines (84).

The rest of the chapter is organized as follows: in section 9.2 we introduce the reader to some of the fundamental principles underlying statistical machine learning, compare them to those that govern more traditional statistical approaches and give some examples. Then, we describe usual candidate mathematical criteria that lead to insurance premium estimation in section 9.3. Statistical learning methodologies are described in section 9.4, with an emphasis on the one that was used within the course of the study. This is followed in section 9.5 by a review of the statistical learning algorithms that we considered, including our best-performing mixture of positive-output

Neural Networks. We describe experimental results with respect to ratemaking in section 9.6. In section 9.7, we compare two models on the task of identifying the risks to be sent to a risk sharing pool facility. In view of these results we conclude with an examination of the prospects for applying statistical learning algorithms to insurance modeling in section 9.8.

9.2 Concepts of Statistical Learning Theory

Statistical inference is concerned with the following: Given a collection of empirical data originating from some functional dependency, provide answers to questions that could be answered if that dependency were known. Although elements of statistical inference have existed for more that 200 years (Gauss, Laplace), it is within the last century that the development of methods and their formal analysis began.

There are two philosophically diverging approaches to the above problem. The goal of the first approach is to *identify* the data generating process. For the purpose of this modelling goal, one must have sufficient knowledge of the physical laws that govern the process in order to build a corresponding model. On the other hand, according to the second approach, one merely attemps to *predict* properties of future data, based on the already given observed data. The belief is that the reality of the process is too complex to be identified and captured in a single unifying model.

One crucial element of the evaluation of the generalization ability of a particular model is the measurement of the predictive performance results on *out-of-sample* data, i.e., using a collection of data, disjoint from the *in-sample* data that has already been used for model parameter estimation. In the case of automobile insurance, where data is not *i.i.d.* but rather bears a sequential structure with potential non-stationarities (changes in the underlying generating process with respect to its explanatory variables), this requirement leads to the particular methodology of *sequential validation* which we shall explain in detail in section 9.4.

Another reason for prefering the operational approach in practical applications: **the out-of-sample statistical tests do not require any assumption on the form of the underlying distribution**[1]. In other words,

[1] the only assumption, in ordinary tests, is that the data points are generated *i.i.d.*, independently from the same distribution. Even this assumption can be relaxed in order to deal with sequentially dependent data (72; 27; 16; 18).

when performing a classical parametric test, the conclusion of the test could generally be invalidated if strictly speaking the data was not generated from the presumed class of parametric distributions. When the livelihood of a corporation is at stake in these choices, it might be wiser to avoid relying on such assumptions.

An important dilemma that arises in statistical inference is that of the bias-variance trade-off (39) in generalization error. Choosing a function that is too simple by "smoothing" too much (in the non-parametric statistics sense) or choosing a capacity or class of functions that is too small (in Vapnik's VC-theory (84) sense) both introduce an additional bias and lead to *underfitting*. Conversely, too little "smoothing", a too large capacity or class of functions will lead to increased variance of the estimators and *overfitting*. One should attempt to strike the optimal balance between bias and variance, this is the question that **model selection** algorithms address.

9.3 Mathematical Objectives

The first goal of insurance premium modeling is to estimate the *expected claim amount* for a given insurance contract for a future period (usually one-year). Here we consider that the amount is 0 when no claim is filed. Let $X \in \mathbf{R}^n$ denote the customer and contract *input profile*, a vector representing all the information known about the customer and the proposed insurance policy before the beginning of the contract. Let $A \in \mathbf{R}^+$ denote the amount that the customer claims during the contract period; we shall assume that A is non-negative. Our objective is to estimate this claim amount, which is the *pure premium* p_{pure} of a given contract x:[2]

$$p_{pure}(x) = E_A[A|X = x]. \tag{9.1}$$

where $E_A[\cdot]$ denotes expectation, i.e. the average over an infinite population, and $E_A[A|X = x]$ is a conditional expectation, i.e. the average over a subset of an infinite population, comprising only the cases satisfying the condition $X = x$.

[2]The pure premium is distinguished from the premium actually charged to the customer, which must account for the underwriting costs (marketing, commissions, premium tax), administrative overhead, risk and profit loadings and other costs.

9.3.1 The Precision Criterion

In practice, of course, we have no direct access to the quantity (9.1), which we must estimate. One possible criterion is to seek the *most precise* predictor, which minimizes the expected squared error (ESE) over the unknown distribution:

$$E_{A,X}[(p(X) - A)^2], \tag{9.2}$$

where $p(X)$ is a pure premium predictor and the expectation is taken over the random variables X (input profile) and A (total claim amount). Since $P(A, X)$, the true joint distribution of A and X, is unknown, we can **unbiasedly** estimate the ESE performance of an estimator p on a data set $D_{test} = \{\langle x_i, a_i \rangle\}_{i=1}^{N}$, as long as this data set is not used to choose p. We do so by using the **mean squared error** on that data set:

$$\frac{1}{N} \sum_{\langle x_i, a_i \rangle \in D_{test}} (p(x_i; \theta) - a_i)^2, \tag{9.3}$$

where θ is the vector of parameters of the model used to compute the premiums. The vector x_i represents the i^{th} input profile of dataset D_{test} and a_i is the claim amount associated to that input profile. Thus, D_{test} is a set of N insurance policies. For each policy, D_{test} holds the input profile and associated incurred amount.

Algorithms that try to minimize the expected squared error simultaneously reduce both the bias and the variance of the estimators, striking a tradeoff that minimizes the sum of both (since the remainder is the noise, which cannot be reduced by the choice of predictor). On the other hand, with a rule such as minimum bias used with table based methods, cells are merged up to a point where each cell has sufficient credibility, i.e., where the variance is sufficiently low. Then, once the credibility (and variance) level is set fixed, the bias is minimized. On the contrary, by targeting minimization of the expected squared error one avoids this arbitrary setting of a credibility level.

In comparison to parametric approaches, this approach avoids distributional assumptions. Furthermore, it looks for an optimal trade-off between bias and variance, whereas parametric approaches typically focus on the unbiased estimators (within a class that is associated with a certain variance). Because of the above trade-off possibility, it is always possible (with a finite data set) to improve an unbiased estimator by trading a bit of bias increase for a lot of variance reduction.

9.3.2 The Fairness Criterion

In insurance policy pricing, the precision criterion is not the sole part of the picture; just as important is that the estimated premiums do not systematically discriminate against specific segments of the population. We call this objective the *fairness criterion*, sometimes referred to as *actuarial fairness*. We define the *bias of the premium* $b(P)$ to be the difference between the average pure premium and the average incurred amount, in a given sub-population P of dataset D:

$$b(P) = \frac{1}{|P|} \sum_{\langle x_i, a_i \rangle \in P} p(x_i) - a_i, \tag{9.4}$$

where $|P|$ denotes the cardinality of the sub-population P, and $p(\cdot)$ is some premium estimation function. The vector x_i represents the i^{th} input profile of sub-population P and a_i is the claim amount associated to that input profile. A possible fairness criterion would be based on minimizing the sum, over a certain set of critical sub-populations $\{P_k\}$ of dataset D, of the square of the biases:

$$\sum_{k, P_k \in D} b^2(P_k) \tag{9.5}$$

In the particular case where one considers all sub-populations, then both the fairness and precision criterions yield the same optimal solution, i.e., they are minimized when $p(x_i) = E[A_i|x_i]$, $\forall i$, i.e., for every insurance policy, the premium is equal to the conditional expectation of the claim amount. The proof is given in section 9.9.

In order to measure the fairness criterion, we used the following methodology: after training a model to minimize the MSE criterion (9.3), we define a finite number of disjoint subsets (sub-populations) of test set D: $P_k \subset D, P_k \cap P_{j \neq k} = \emptyset$, and *verify* that the absolute bias is not significantly different from zero. The subsets P_k can be chosen at convenience; in our experiments, we considered 10 subsets of equal size delimited by the deciles of the test set premium distribution. In this way, we verify that, for example, for the group of contracts with a premium between the 5th and the 6th decile, the average premium matches, within statistical significance, the average claim amount.

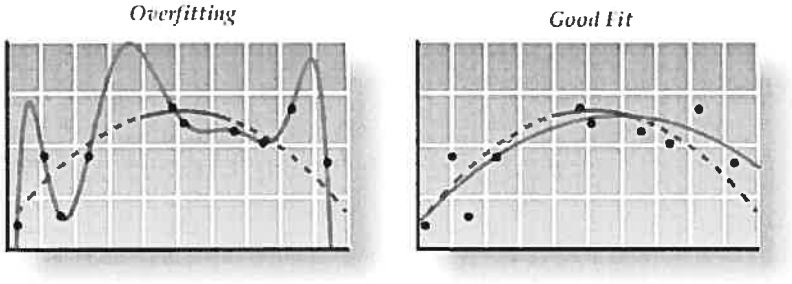

Figure 9.1: Illustration of **overfitting**. The solid left curve fits the noise in the data points (black dots) and has not learned the underlying structure (dashed). The right curve, with less flexibility, does not overfit.

9.4 Methodology

A delicate problem to guard against when applying statistical learning algorithms is that of *overfitting*. It has precisely to do with striking the right trade-off between bias and variance (as introduced in the previous section), and is known in technical terms as *capacity control*. Figure 9.1 illustrates the problem: the two plots show empirical data points (black dots) that we are trying to approximate with a function (solid red curve). All points are sampled from the same underlying function (dashed blue curve), but are corrupted with noise; the dashed curve may be seen as the "true" function we are seeking to estimate.

The left plot shows the result of fitting a very flexible function, i.e. a high-order polynomial in this case, to the available data points. We see that the function fits the data points perfectly: there is zero error (distance) between the red curve and each of the black dots. However, the function oscillates wildly between those points; it has not captured any of the fundamental features of the underlying function. What is happening here is that the function has mostly *captured the noise* in the data: it overfits.

The right plot, on the other hand, shows the fitting of a less flexible function, i.e. a 2nd-order polynomial, which exhibits a small error with respect to each data point. However, by not fitting the noise (because it does not have the necessary degrees of freedom), the fitted function far better conveys the structural essence of the matter.

Capacity control lies at the heart of a sound methodology for data mining

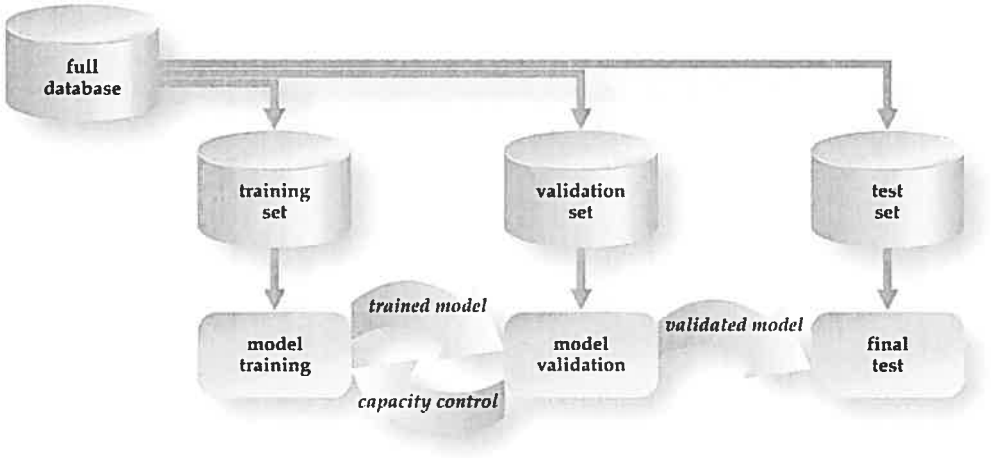

Figure 9.2: Methodology to prevent overfitting. Model capacity is controlled via a validation set, disjoint from the training set. The generalization performance estimator is obtained by final testing on the test set, disjoint from the first two.

and statistical learning algorithms. The goal is simple: to choose a function class flexible enough (with enough capacity) to express a desired solution, but constrained enough that it does not fit the noise in the data points. In other words, we want to avoid overfitting **and** underfitting.

Figure 9.2 illustrates the basic steps that are commonly taken to resolve this issue: these are not the only means to prevent overfitting, but are the simplest to understand and use.

1. The full data set is randomly split into three *disjoint* subsets, respectively called the training, validation, and test sets.

2. The training set is used to fit a model with a chosen initial capacity.

3. The validation set is used to *evaluate the performance* of that fitted function, on **different data points** than used for the fitting. The key here is that a function overfitting the training set will exhibit a low performance on the validation set, if it does not capture the underlying structure of the problem.

4. Depending on the validation set performance, the capacity of the model is adjusted (increased or reduced), and a new training phase (step

2) is attempted. This training–validation cycle is repeated multiple times and the capacity that provides the best validation performance is chosen.

5. Finally, the performance of the "ultimate" function (that coming out of the validation phase) is evaluated on data points never used previously —those in the test set—to give a completely unbiased measure of the performance that can be expected when the system is deployed in the field. This is called **generalization performance**.

In the case of automobile insurance, there is a sequential structure that must be respected. When using data from previous years to simulate the out-of-sample performance of models, one should try to replicate as closely as possible the actual process that will be followed to deploy a model. A sound procedure is to split the data according to their policy date year into the training (year $y-3$), validation (year $y-2$) and test (year y) sets. The reason for skipping year $y-1$ is to recognize the fact that at time y, as the model is deployed, year $y-1$ is not yet available. Reducing this gap to a few months would help the insurer account for more recent data and very likely obtain better performance. An insurer having access to data dating from 1995 could obtain test performances for years 1998 and above. Assuming 2002 results are available yields 5 estimates of the test performance of the whole modelling procedure.

9.5 Models

In this section, we describe various models that have been implemented and used for the purpose of ratemaking. For benchmark evaluation purposes, we implemented the constant model. This consists of simply charging every single insurance policy a flat premium, regardless of the associated variable values. The premium is the mean of all incurred amounts as it is the constant value that minimizes the mean-squared error.

$$p(x) = \beta_0, \tag{9.6}$$

where β_0 is the mean and the premium $p(x)$ is independent of the input profile x.

We implemented linear models parameterized by a set of coefficients, one for each variable plus an intercept value:

$$p(x) = \beta_0 + \sum_{i=1}^{n} \beta_i x_i. \tag{9.7}$$

There are two main ways to control the capacity of linear models when in presence of noisy data: (1) using a subset of input variables; this directly reduces the number of coefficients (but choosing the best subset introduces another level of choice which is sometimes detrimental). (2) penalizing the norm of the parameters (in general excluding the intercept β_0); this is called *ridge regression* in statistics, and *weight decay* in the Neural Networks community. This was the main method used to control capacity of the linear model in our experiments.

It should be noted that the premium computed with the linear model can be negative (and negative values are indeed sometimes obtained with the trained linear models). This may happen even if there are no negative amounts in the data, simply because the model has no built-in positivity constraint (unlike the GLM and the softplus Neural Network described below).

9.5.1 Table-based methods

These more traditional ratemaking methods rely mainly on a classification system, base rates and relativities. The target function is approximated by constants over regular (finite) intervals. This gives rise to a typical staircase-like function, where each level of the staircase is given by the value in the corresponding cell in the table. A common refinement in one dimension is to perform a linear interpolation between neighboring cells, to smooth the resulting function somewhat. The table is not limited to two variables; however, when adding a new variable (dimension), the number of cells increases by a factor equal to the number of discretization steps in the new variable.

Assuming that the i^{th} variable of profile x belongs to the j^{th} category, we obtain,

$$p(x) = \beta_0 \prod_{i=1}^{m} \beta_{i,j} + \sum_{i=m+1}^{n} \beta_{i.j}, \tag{9.8}$$

where $\beta_{i,j}$ is the relativity for the j^{th} category of the i^{th} variable and β_0 is

the standard premium. We consider the case where the first m factors are multiplicative and the last $n - m$ factors are additive.

The formula above assumes that all variables have been analyzed individually and independently. A great deal of effort is often put in trying to capture dependencies (or interactions) between some variables and to encode them into the premium model.

An extension of the above is to multiplicatively combine multiple tables associated to different subsets of variables. This is in effect a particular form of Generalized Linear Model (see below), where each table represents the interdependence effects between some variables.

Greedy multiplicative models can be seen as a special case of table-based methods. The basic idea of the Greedy Multiplicative Model is to add one of these multiplicative coefficients at a time. At each step, we have to choose one among the input variables. We choose the variable which would reduce most the training MSE. The coefficient for that component is easily obtained analytically by minimizing the MSE when all the previously obtained coefficients are kept fixed.

In the tables we use the short-hand name "CondMean" for this model because it estimates and combines many conditional means. Note that like the GLM, this model provides positive premiums. These models are commonly used in current practice and they can be viewed as an extension of multiplative table-based methods. .

9.5.2 Generalized Linear Model

The Generalized Linear Models (GLM) were introduced to the actuarial community in the 60's (4). More recently, (15; 49; 69) some experiments have been conducted using such models. We used the exponential function as the link function of the models:

$$p(x) = \exp\left(\beta_0 + \sum_{i=1}^{n} \beta_i x_i\right), \tag{9.9}$$

where, the exponentiation ensures that the resulting premiums are all positive.

The capacity of a GLM model can be controlled using the same techniques as those mentionned above in the context of linear models. Again, note that the GLM always provides a positive premium.

9.5.3 Decision trees

Decision trees such as CHAID (Chi-square Automatic Interaction Detector) split the variable space in smaller subspaces. Any input profile x fits into one and only one of those subspaces called *leaves*. To each leaf is associated a different premium level,

$$p(x) = \sum_{i=1}^{n_l} \mathbf{I}_{\{x \in l_i\}} \beta_i, \tag{9.10}$$

where $\mathbf{I}_{\{x \in l_i\}}$ is an indicator function equal to 1 if and only if x belongs to the i^{th} leaf. In that case, $\mathbf{I}_{\{x \in l_i\}} = 1$ and $p(x) = \beta_i$. Otherwise, $\mathbf{I}_{\{x \in l_i\}}$ is equal to zero, meaning x belongs to another leaf. The number of leaves is n_l. The premium level β_i is set equal to the average incurred amount of the policies for which the profile x belongs to the i^{th} leaf.

The basic way in which capacity is controlled is through several hyper-parameters: minimum population in each leaf, minimum population to consider splitting a node, maximum height of the decision tree and Chi-square statistic threshold value.

Within each leaf, one can replace the associated constant premium value with a linear regression. Each leaf then has its own set of regression coefficients. There are thus n_l different linear regressions of $n + 1$ coefficients each.

$$p(x) = \sum_{i=1}^{n_l} \mathbf{I}_{\{x \in l_i\}} \left(\beta_{i,0} + \sum_{j=1}^{n} \beta_{i,j} x_j \right). \tag{9.11}$$

Each linear regression was fit to minimize the mean-squared error on the training cases that belong to its leaf. For reasons that are clear in the light of learning theory, a tree used in such a combination should have less leaves than an ordinary CHAID tree. In our experiments we have chosen the size of the tree based on the validation set MSE. The linear models were computed after the tree structure was defined.

In these models, capacity is controlled with the same hyper-parameters as CHAID, and there is also the question of finding the right *weight decay* for the linear regression. Again, the validation set is used for this purpose.

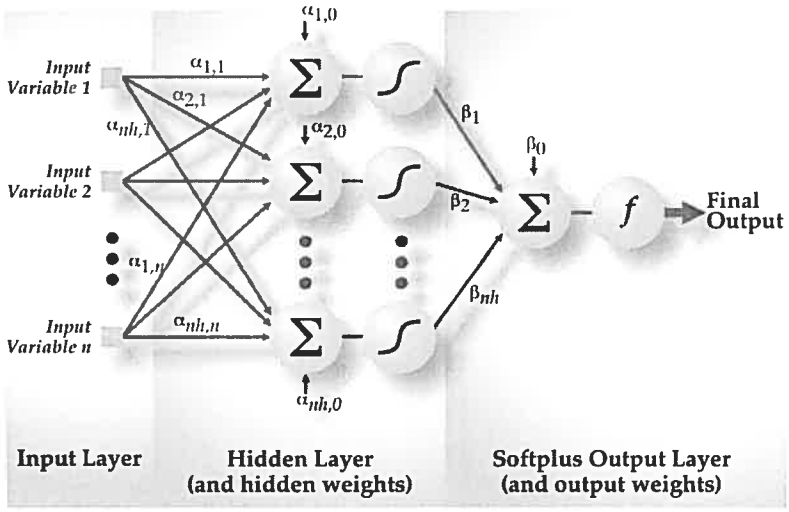

Figure 9.3: Topology of a one-hidden-layer Neural Network. In each unit of the hidden layer, the variables are linearly combined. The network then applies a non-linear transformation on those linear combinations. Finally, the resulting values of the hidden units are linearly combined in the output layer.

9.5.4 Neural Networks

Figure 9.3 illustrates a typical *multi-layer feedforward architecture* such as the ones that were used for the current project and for which the premium is computed as such:

$$p(x) = \beta_0 + \sum_{i=1}^{n_h} \beta_i \tanh\left(\alpha_{i,0} + \sum_{j=1}^{n} \alpha_{i,j} x_j\right). \tag{9.12}$$

The number of hidden units (n_h above) plays a crucial role in our desire to control the capacity of the Neural Network as it is proportional to the number of parameters used by the model. Another technique for controlling the capacity of a Neural Network is to use weight decay, i.e., a penalized training criterion that limits the size of the parameters of the Neural Network.

Note that like the linear regression, this model can potentially yield negative premiums in some cases. We have observed much fewer such cases than with the linear regression.

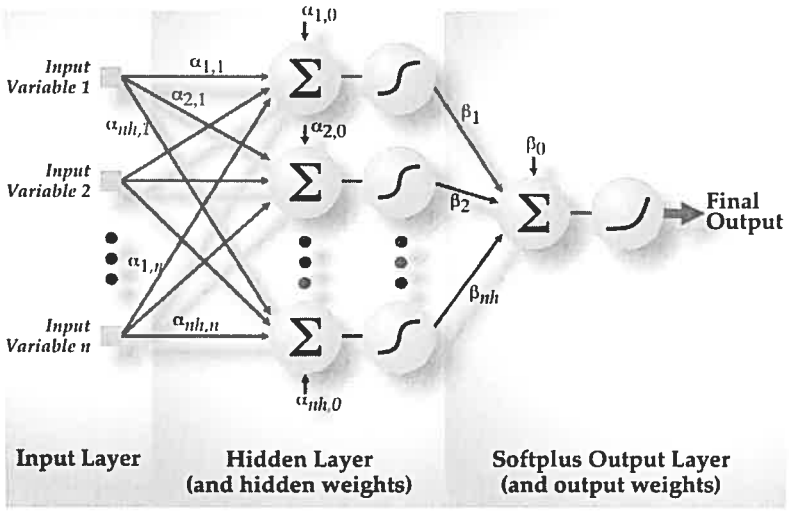

Figure 9.4: Topology of a one-hidden-layer softplus Neural Network. The hidden layer applies a non-linear transformation of the variables, whose results are linearly combined by the output layer. The softplus output function forces the function to be positive. To avoid cluttering, some weights linking the variables to the hidden layer are omitted on the figure.

The softplus function was recently introduced (29) as a means to model a convex relationship between an output and one of its inputs. We modified the Neural Network architecture and included a softplus unit as a final transfer function. Figure 9.4 illustrates this new architecture we have introduced for the purpose of computing insurance premiums. The corresponding formula is as such:

$$p(x) \quad = \quad F\left(\beta_0 + \sum_{i=1}^{n_h} \beta_i \ \tanh\left(\alpha_{i,0} + \sum_{j=1}^{n} \alpha_{i,j} x_j\right)\right), \tag{9.13}$$

where $F(\cdot)$ is the softplus function which is actually simply the primitive (integral) function of the "sigmoid" function. Thus

$$F(y) \quad = \quad \log\left(1 + e^{y}\right). \tag{9.14}$$

The softplus function is convex and monotone increasing with respect to its input and always strictly positive. Thus, this proposed architecture leads to strictly positive premiums.

In preliminary experiments we have also tried to use the exponential function (rather than the softplus function) as the final transfer function. However we obtained poor results due to difficulties in the optimization (probably due to the very large gradients obtained when the argument of the exponential is large). The capacity of the softplus Neural Network is tuned just like that of an ordinary Neural Network.

9.5.5 Regression Support Vector Machine

Support Vector Machines (SVM) have recently been introduced as a very powerful set of non-parametric statistical learning algorithms (84; 78). They have been very successful in classification tasks, but the framework has also been extended to perform regression. Like other *kernel methods* the class of functions has the following form:

$$p(x) = \sum_i \alpha_i K(x, x_i) \tag{9.15}$$

where x_i is the input profile associated with one of the training records, and α_i is a scalar coefficient that is learned by the algorithm and K is a *kernel function* that satisfies the Mercer condition (22):

$$\int_C K(x,y)\, g(x)\, g(y)\, dx\, dy \geq 0 \tag{9.16}$$

for any square integrable function $g(x)$ and compact subset C of $\mathbf{R}^n$. This Mercer condition ensures that the kernel function can be represented as a simple dot product:

$$K(x,y) = \phi(x) \cdot \phi(y) \tag{9.17}$$

where $\phi()$ is a function that projects the input profile vector into a (usually very) high-dimensional "feature" space, usually in a nonlinear fashion. This leads us, to a simple expression for the premium function:

$$\begin{aligned} p(x) &= \sum_i \alpha_i \phi(x) \cdot \phi(x_i) \\ &= \phi(x) \cdot \left(\sum_i \alpha_i \phi(x_i) \right) \\ &= w \cdot \phi(x). \end{aligned} \tag{9.18}$$

Thus, in order to compute the premium, one needs to project input profile x in its feature space and compute a dot product with vector w. This vector w depends only on a certain number of input profiles from the training dataset and their associated coefficients. These input profiles are referred to as the *support vectors* and have been selected, along with their associated coefficients by the optimization algorithm.

SVMs have several very attractive theoretical properties, including the fact that an exact solution to the optimization problem of minimizing the training criterion can be found, and the capacity of the model is automatically determined from the training data. In many applications, we also find that most of the α_i coefficients are zero.

However, in the case of insurance data, an important characteristic of regression SVMs is that they are NOT trained to minimize the training MSE. Instead they minimize the following criterion:

$$J = \frac{1}{2}||w||^2 + \lambda \sum_i |a_t - p(x_i)|_\epsilon \tag{9.19}$$

where $|e|_\epsilon = \max(0, |e| - \epsilon)$, λ and ϵ trade-off accuracy with complexity, a_i is the observed incurred claim amount for record i, x_i is the input profile for record i, and the vector w is defined in terms of the α_i coefficients above. It can therefore be seen that this algorithm minimizes something close to the *absolute value of the error* rather than the *squared error*. As a consequence, the SVM tends to find a solution that is close to the *conditional median* rather than the *conditional expectation*, the latter being what we want to evaluate in order to set the proper value for a premium. Furthermore, note that the insurance data display a highly asymmetric distribution, so the median and the mean are very different. In fact, the conditional median is often exactly zero. Capacity is controlled through the ϵ and λ coefficients.

9.5.6 Mixture Models

Mixture of experts belong to the larger class of learning algorithms called "committee machines" in which an ensemble of estimators are trained and the prediction of the committee for a new input is generated in form of a combination of the predictions of individual committee members. Committee machines differ from one another in the way they weight different inputs.

In *simple averaging*, the predictions of the committee members are given equal weight. This can be useful for example when committee members are

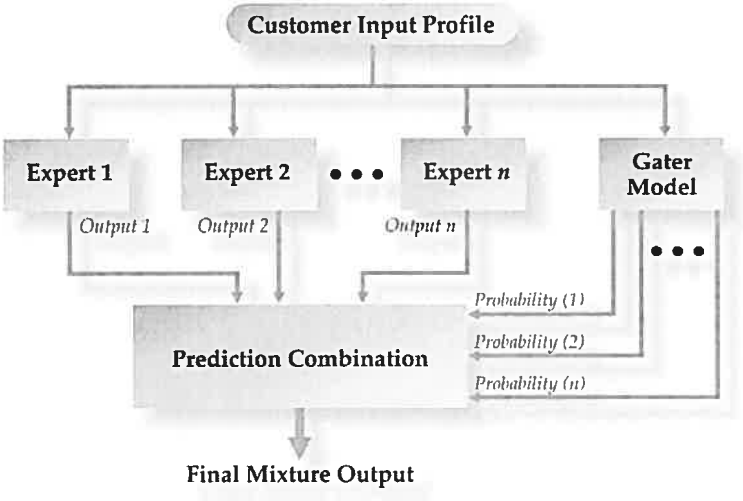

Figure 9.5: Schematic representation of the mixture model. The first-stage models each make an independent decision, which are linearly combined by a second-stage **gater**.

neural networks as a decorrelation among the predictions of the members can be achieved simply by varying the initial conditions (weights) in training the neural networks such that they converge to different local minima. In *bagging* ("boostrap aggregating") (13), a series of bootstrap samples are drawn from the dataset, a member is trained on each of these samples and their predictions are given equal weight. Bagging is particularly useful when members are models of high variance with respect to dataset. In *boosting* (76), the members are trained sequentially, each member paying more attention to the previously misclassified training examples. According to *stacking* (87), the committee members are combined with weights that depend on their cross-validation performance. Thus simple averaging and bagging (in its simplest form) equally weight each member's outputs whereas boosting and stacking weight them according to an estimate of the members' generalization performance.

The *mixture of experts* has been proposed (60) in order to decompose the learning problem, the initial motivation being to design a system in which

different neural networks are responsible for modeling different regions of the input space. This modularity leads to greater modeling capability and potentially to a meaningful and interpretable segmentation of the map. The mixture of experts expresses the conditional as a linear combination of the predictions of the members (or *expert models*), with weights determined by a *gater model*. The experts are specialized predictors that each estimate the pure premium for insureds that belong to a certain class. The *gater* attempts to predict to which class each insured belongs, with an estimator of the conditional probability of the class given the insured's input profile. For a mixture model, the premium can be expressed as

$$p(x) = \sum_i p(i|x)p_i(x) \tag{9.20}$$

where $p(i|x)$ is the probability that the expert i is selected given an insured with input profile x. This value is determined by the gater model. Also, $p_i(x)$ is the premium, as computed by the expert model of class i, associated to input profile x.

A trivial case occurs when expert i is deterministically found for any particular input profile x, i.e., each expert is assigned a region of the input space and these regions are non-overlapping. In that case, we simply split the training database and train each expert model on a subset of the data. The gater then simply assigns a value of $p(i|x) = 1$ if i is the appropriate class for input profile x and zero otherwise. This is in fact fundamentally equivalent to other techniques such as decision trees or table-based methods.

A more general and powerful approach is to have the learning algorithm discover a relevant probabilistic decomposition of the data into different regions of the input space which then become the classes and are encoded in the gater model. In that case, both the gater and the experts are trained together. The usual probabilistic setting for the mixture of expert is as such: given an input x, expert i is selected with probability $p(i|x)$. An output is generated according to probability density function $f(y|i,x)$ with mean value $p_i(x)$. Since it is unknown in the data which expert produced the output, the probability density function of an output given the input is a mixture distribution:

$$f(y|x) = \sum_{i=1}^{M} p(i|x)f(y|i,x) \tag{9.21}$$

The mixture we propose in this study is built on the assumption that *experts have non-overlapping output distributions.* Accordingly, once the variable y is fixed, the variable i is degenerated, i.e., the expert can be unambiguously identified. A first advantage of this assumption is that, once the domains of expertise in output space have been determined, experts can be trained using only a subset of the data, therefore accelerating the training of the experts.

In the usual mixture of experts setting, each expert is trained to model the stochastic process for a region of the input space. This may prove efficient if the gater model is able to identify regions of the input space such that the process varies significantly across the different regions. The hope is the reduction in bias of having experts each assigned a subtask will compensate for the increased variance since each of them will be trained with a subset of the entire dataset.

The proposed mixture of experts takes a different approach by assuming the regions of interest for splitting the learning task into subtasks can be identified in output space. Rather than having each expert assigned a region of the input space and learning the stochastic process for that region, we assign each expert to a region of output space and let the gater model determine the probability for each region of the output space, that an input profile will generate an output in that specific region.

Whereas splitting the input space into non-overlapping regions leads to a trivial gater model, the proposed mixture does not alleviate the task of developping a probabilistic gater model: as a new example is presented, the output values are unknown and the example cannot be assigned to an expert deterministically.

Splitting the output space into regions can prove efficient if the process varies significantly across regions. In this study, we used automobile insurance data with an asymmetric fat-tailed distribution of claims. Also, most insureds do not file any claim. We can therefore easily identify three regions: no claims, small claims and large claims. Most information lies in the region of small claims. Large claims represent mainly noise. Profiles without a claim bring little information about the probability of making a claim and none as to the severity distribution of claims.

In this study both the experts and the gater are softplus Neural Networks, but any other model can be used. In Figure 9.5, we schematically illustrate a mixture model as the one that was used in the framework of this project. Note that the proposed approach can be viewed as a generalization of the

Table 9.1: Comparison between the main models, with MSE on the training set, validation set, and test sets. The MSE is with respect to claim amounts and premiums expressed in thousand of dollars.

Model	Train MSE	Valid MSE	Test MSE
Constant	56.1108	56.5744	67.1192
Linear	56.0780	56.5463	67.0909
GLM	56.0762	56.5498	67.0926
NN	56.0706	56.5468	67.0903
Softplus	56.0704	56.5480	67.0918
CHAID	56.0917	56.5657	67.1078
CondMean	56.0827	56.5508	67.0964
Mixture	56.0743	**56.5416**	**67.0851**

frequency/severity approach. Consider the frequency model as the gater and the severity model as one of the experts. In our case, we split the severity model in two: low severity model and high severity model. This puts the increased burden on the gater model model to compute the probabilities, given an input profile, of "choosing" the low severity model or the high severity model. The high and low severity models are trained on less data but we expect them to have lower bias.

9.6 Experimental Results

9.6.1 Mean-Squared Error Comparisons

Table 9.1 summarizes the main results concerning the comparison between different types of statistical machine learning algorithms. All the models have been trained using the same input profile variables. For each insurance policy, a total of 33 input variables were used and the total claims for an accident came from five main coverages: bodily injury, accident benefit, property damage, collision and comprehensive. Two other minor coverages were also included: death benefit and loss of use. In the table, *NN* stands for Neural Network, *GLM* for Generalized Linear Model, and *CondMean* for the Greedy Multiplicative Model. The MSE on the training set, validation set and test

set are shown for all models. The MSE is with respect to claim amounts and premiums expressed in **thousand of dollars**. The model with the lowest MSE is the "Mixture model", and it is the model that has been selected for the comparisons with the insurer's current rules for determining insurance premiums to which we shall refer as the *Rule-Based Model.*

One may wonder from the previous table why the MSE values are so similar across various models for each dataset and much different across the datasets. In particular, all models perform much worse on the testset (in terms of their MSE). There is a very simple explanation. The maximum incurred amount on the test set and on the validation set is around 3 million dollars. If there was one more such large claim in the test set than in the validation set, one would expect the test MSE (calculated for premiums and amounts in thousand of dollars) to be larger by about 7 (these are in units of squared thousand dollars). Thus a difference of 11 can easily be explained by a couple of large claims. This is a reflection of the very thick right-hand tail of the incurred amount distribution (whose standard deviation is only of about 8 thousand dollars). Conversely, this also explains why all MSE are very similar across models for one particular dataset. The MSE values are all mainly driven by very large claims which no model could reliably forecast (no model could lead the insurer to charge a million dollars to a particular insured !) Consequently, truely significant differences between model performances are shadowed by the effect of very large claims on the MSE values. Although the differences between model performance are relatively small, we shall see next that careful statistical analysis allows us to discover that some of them are significant.

Figure 9.6 illustrates graphically the results of the table, with the models ordered according to the validation set MSE. One should note that within each class of models the capacity is tuned according to the performance on the validation set. On the test and validation sets, the Mixture model dominates all the others. Then come the ordinary Neural Network, linear model, and softplus Neural Network. Only slightly worse are the GLM and CondMean (the Greedy Multiplicative model). CHAID fared poorly on this dataset. Note that the CHAID tree with linear models in the leaves, as described above, performed worse than ordinary CHAID. Finally, the constant model is shown as a baseline (since it corresponds to assigning the same premium to every 1-year policy). It is also interesting to note from the figure that the model with the lowest training MSE is not necessarily the best out-of-sample (on the validation or test sets). The SVM performance was appalling and

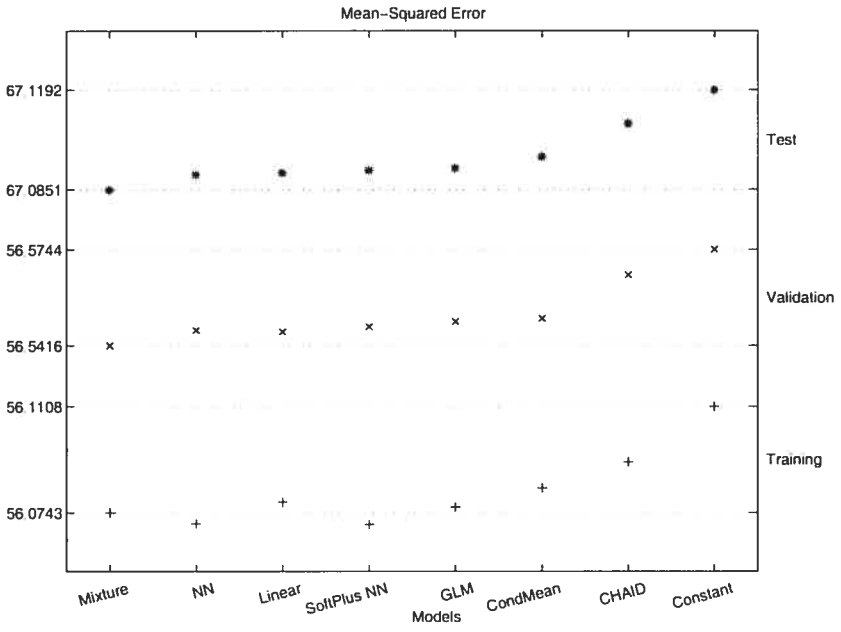

Figure 9.6: MSE results (from table 9.1) for eight models . Models have been sorted in ascending order of test results. The training, validation and test curves have been shifted closer together for visualization purposes. The out-of-sample test performance of the mixture model is significantly better than any of the other. Validation based model selection is confirmed on test results.

is not shown here; it did much worse than the constant model, because it is aiming for the conditional median rather the conditional expectation, which are very different for this kind of data.

Table 9.2 shows a statistical analysis to determine whether the differences in MSE between the Mixture model and each of the other models are significant. We used the Diebold-Marino test (66) whereby in order to compare models A and B, we compute, for each profile of the test set, the following statistic:

$$dm_i = (p_A(x_i) - a_i)^2 - (p_B(x_i) - a_i)^2 \tag{9.22}$$

where a_i is the claim amount for profile x_i. We then compute the mean and variance of the dm_i statistics and conduct a standard test.

Table 9.2: Statistical Comparison Between Different Learning Models and the Mixture Model. The p-value is for the null hypothesis of no difference between Model #1 and the best mixture model. Symbols $\hat{\mu}$ and $\hat{\sigma}$ stand for sample mean and standard deviation. Note that ALL differences are statistically significant.

Model #1	Model #2	$\hat{\mu}$	$\hat{\sigma}$	Z	p-value
Constant	Mixture	3.41e-02	3.33e-03	10.24	**0**
Linear	Mixture	5.82e-03	1.32e-03	4.41	**5.30e-06**
GLM	Mixture	7.54e-03	1.15e-03	6.56	**2.77e-11**
NN	Mixture	5.24e-03	1.41e-03	3.71	**1.03e-04**
Softplus	Mixture	6.71e-03	1.09e-03	6.14	**4.21e-10**
CHAID	Mixture	2.36e-02	2.58e-03	9.15	**0**

The *Mean* column shows the difference in MSE with the Mixture model. The next column shows the *Standard Error* of that mean. Dividing the mean by the standard error gives Z in the next column. The last column gives the *p-value* of the null hypothesis according to which the true expected squared errors for both models are the same. Conventionally, a value below 5% or 1% is interpreted as indicating a significant difference between the two models. The p-values and Z corresponding to significant differences are highlighted. Therefore the differences in performance between the mixture and the other models are all statistically significant. As mentionned above, the MSE values are very much affected by large claims. Does such a sensitivity to very large claims make statistical comparisons between models incorrect? No. Fortunately all the comparisons are performed on **paired data** (the squared error for each individual policy), which cancel out the effect of these very large claims (since, for these special cases, the squared error will be huge for all models and of very close magnitude)

Table 9.3 has similar columns, but it provides a comparison of pairs of models, where the pairs are consecutive models in the order of validation set MSE. What can be seen is that the ordinary Neural Network (NN) is significantly better than the linear model, but the latter, the softplus Neural Network and GLM are not statistically distinguishable. Finally GLM is significantly better than CHAID, which is significantly better than the constant

Table 9.3: Statistical Comparison Between Pairs of Learning Models. Models are ordered from worst to best. Symbols $\hat{\mu}$ and $\hat{\sigma}$ stand for sample mean and standard deviation. The test is for comparing the sum of MSEs. The p-value is for the null hypothesis of no difference between Model #1 and Model #2.

Model #1	Model #2	$\hat{\mu}$	$\hat{\sigma}$	Z	p-value
Constant	CHAID	1.05e-02	2.62e-03	**3.99**	**3.24e-05**
CHAID	GLM	1.60e-02	2.15e-03	**7.46**	**4.23e-14**
GLM	Softplus	8.29e-04	8.95e-04	0.93	1.77e-01
Softplus	Linear	8.87e-04	1.09e-03	0.82	2.07e-01
Linear	NN	5.85e-04	1.33e-03	0.44	3.30e-01
NN	Mixture	5.23e-03	1.41e-03	**3.71**	**1.03e-04**

model. Note that although the softplus Neural Network alone is not doing very well here, it is doing very well within the Mixture model (it is the most successful one as a component of the mixture). The reason may be that within the mixture, the parameter estimation for model of the low incurred amounts is not polluted by the very large incurred amounts (which are learned in a separate model).

9.6.2 Evaluating Model Fairness

Although measuring the predictive accuracy—as done with the MSE in the previous section—is a useful first step in comparing models, it tells only part of the story. A given model could appear significantly better than its competitors *when averaging over all customers*, and yet perform miserably when restricting attention to a subset of customers.

We consider a model to be *fair* if different cross-sections of the population are not significantly biased against, compared with the overall population. Model fairness implies that the average premiums within each sub-group should be statistically close to the average incurred amount within that sub-group.

Obviously, it is nearly impossible to correct for any imaginable bias since there are *many* different criteria to choose from in order to divide the population into subgroups; for instance, we could split according to any single

variable (e.g. premium charged, gender, rate group, territory) but also *combinations of variables* (e.g. all combinations of gender and territory, etc.). Ultimately, by combining enough variables, we end up identifying individual customers, and give up any hope of statistical reliability.

As a first step towards validating models and ensuring fairness, we choose the subgroups corresponding to the location of the deciles of the premium distribution. The i-th decile of a distribution is the point immediately above $10i\%$ of the individuals of the population. For example, the 9-th decile is the point such that 90% of the population come below it. In other words, the first subgroup contains the 10% of the customers who are given the lowest premiums by the model, the second subgroup contains the range 10%–20%, and so on.

The subgroups corresponding to the Mixture Model (the proposed model) differ slightly from those in the *Rule-Based Model* (the insurer's current rules for determining insurance premiums). Since the premium distribution for both models is not the same. The subgroups used for evaluating each model are given in Table 9.4. Since they correspond to the deciles of a distribution, all the subgroups contain approximately the same number of observations ($\approx 28,000$ on the 1998 test set).

The bias within each subgroup appears in Figure 9.7. It shows the average difference between the premiums and the incurred amounts, within each subgroup (recall that the subgroups are divided according to the premiums charged by each model, as per Table 9.4). A positive difference implies that the average premium within a subgroup is higher than the average incurred amount within the same subgroup. 95% confidence intervals on the mean difference are also given, to assess the statistical significance of the results.

Since subgroups for the two models do not exactly represent the same customers, we shall refrain from directly comparing the two models on a given subgroup. We note the following points:

- For most subgroups, the two models are being fair: the bias is usually not statistically significantly different from zero.

- More rarely, the bias is significantly positive (the models overcharge), but never significantly negative (models undercharge).

- The only subgroup for which both models undercharge is that of the highest-paying customers, the 10-th subgroup. This can be understood,

Table 9.4: Subgroups used for evaluating model fairness, for the Mixture and Rule-Based Models. The lowest and highest premiums in the subgroups are given. Each subgroup contains the same number of observations, $\approx 28,000$.

	Mixture Model		Rule-Based Model	
	Low	High	Low	High
Subgroup 1	50.81	166.24	139.27	245.01
Subgroup 2	166.24	214.10	245.01	297.04
Subgroup 3	214.10	259.74	297.04	336.75
Subgroup 4	259.74	306.26	336.75	378.41
Subgroup 5	306.27	357.18	378.41	417.58
Subgroup 6	357.18	415.93	417.58	460.27
Subgroup 7	415.93	490.34	460.26	507.08
Subgroup 8	490.35	597.14	507.07	554.29
Subgroup 9	597.14	783.90	554.29	617.12
Subgroup 10	783.90	4296.78	617.14	3095.79

as these customers represent the highest risk; a high degree of uncertainty is associated with them. This uncertainty is reflected in the huge confidence intervals on the mean difference, wide enough not to make the bias significantly different from zero in both cases. (The bias for the Rule-Based Model is nearly significant.)

From these results, we conclude that both models are usually fair to customers in all premium subgroups. A different type of analysis could also be pursued, asking a different question: "In which cases do the Mixture and the Rule-Based Models differ the most?" We address this issue in next section.

9.6.3 Comparison with Current Premiums

For this comparison, we used the best (on the validation set) *Mixture model* and compared it on the test data of 1998 against the insurer's Rule-Based Model. Note that for legislative reasons, the Rule-Based Model did not use the same variables as the proposed Mixture Model.

Histograms comparing the distribution of the premiums between the

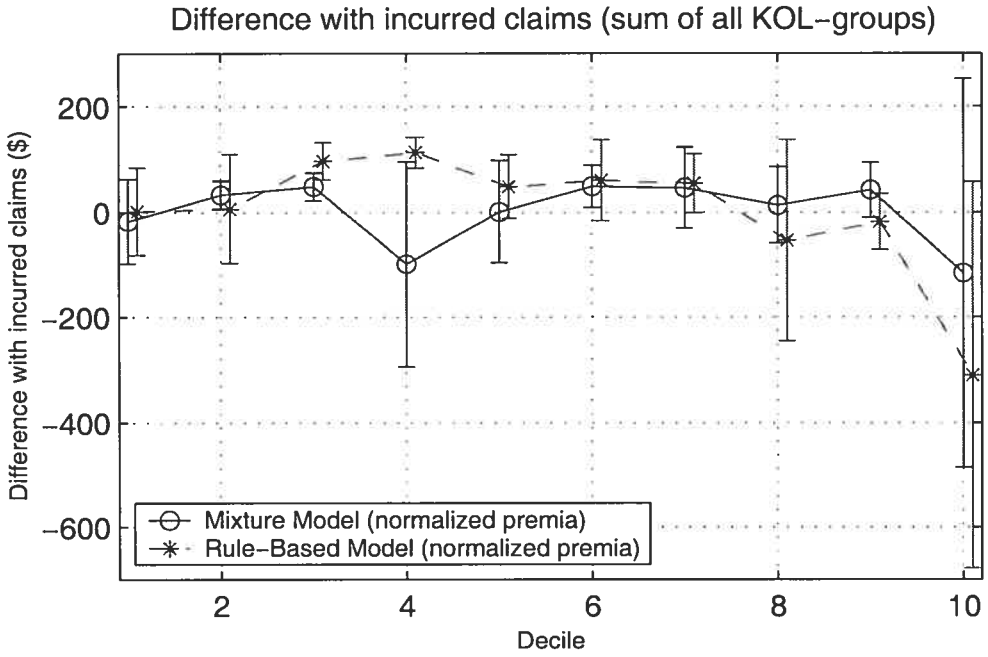

Figure 9.7: Average difference between premiums and incurred amounts (on the sum over all coverage groups), for the Mixture and Rule-Based models, for each decile of the models' respective premium distribution. We observe that both models are being fair to most customers, except those in the last decile, the highest-risk customers, where they appear to under-charge. The error bars represent 95% confidence intervals. (Each decile contains $\approx 28,000$ observations.)

Rule-Based and the Mixture models appear in Figure 9.8. We observe that the premiums from the Mixture model are smoother and exhibit fatter tails (more probability mass in the right-hand side of the distribution, far from the mean). The Mixture model is better able to recognize risky customers and impose an appropriately-priced premium.

This observation is confirmed by looking at the distribution of the *premium difference* between the Rule-Based and Mixture models, as shown in Figure 9.9.

We note that this distribution is extremely skewed to the left. This means that for some customers, the Rule-Based model considerably under-charges

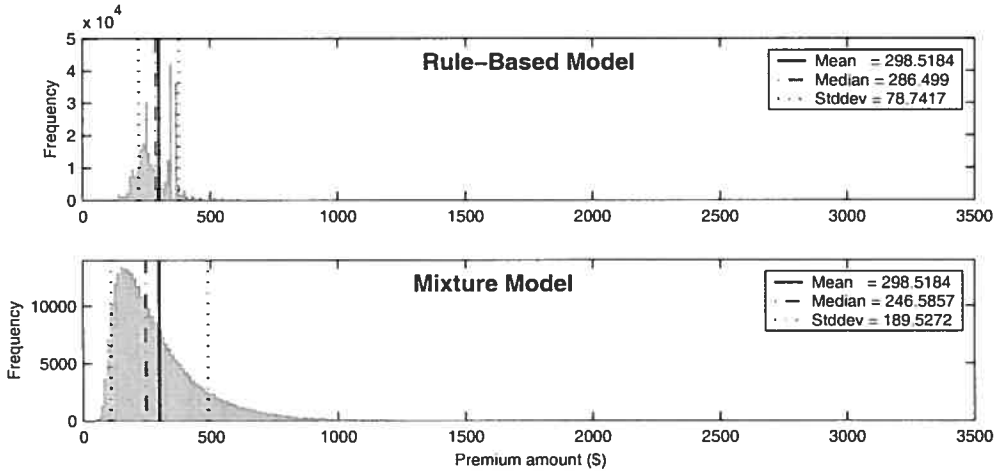

Figure 9.8: Comparison of the premium distribution for the current Rule-Based model and the Mixture model. The distributions are normalized to the same mean. The Mixture model distribution has fatter tails and is much smoother.

with respect to the Mixture model. Yet, the median of the distribution is above zero, meaning that the *typical customer pays more under the Rule-Based model than under the Mixture model.* At the same time, the Mixture model achieves better prediction accuracy, as measured by the *Mean-Squared Error* (MSE) of the respective models, all the while remaining fair to customers in all categories.

Our overriding conclusion can be stated plainly: the Mixture model correctly charges less for typical customers, and correctly charges more for the "risky" ones. This may be due in part to the use of more variables, and in part to the use of a statistical learning algorithm which is better suited to capturing the dependencies between many variables.

9.7 Application to Risk Sharing Pool Facilities

In some provinces and states, improved discrimination between good and bad risks can be used for the purpose of choosing the insureds to be ceeded to risk-sharing pool facilities. In this section, we illustrate the performance of

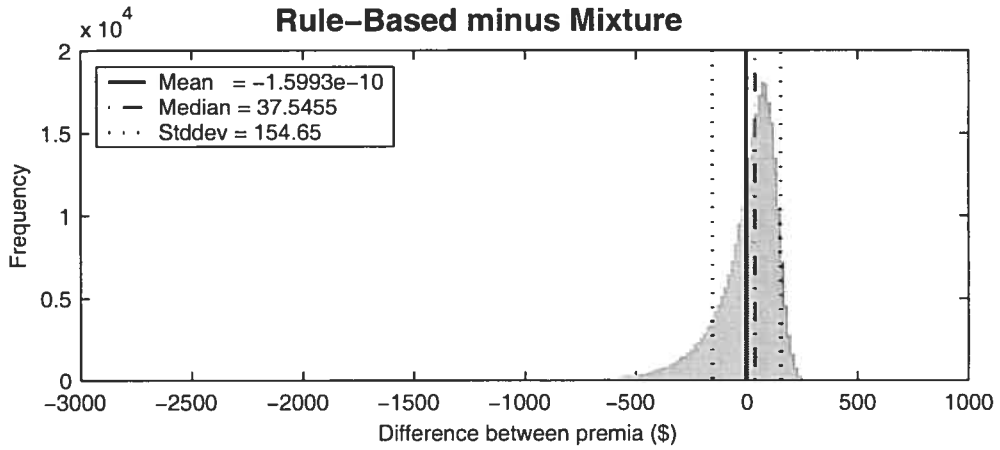

Figure 9.9: Distribution of the premium difference between the Rule-Based and Mixture models, for the sum of the first three coverage groups. The distribution is negatively skewed.

some algorithms when applied to this feat according to the rules that apply in Québec *Plan de Répartition des Risques* (PRR). According to this facility, an insurer can choose to ceed up to 10% of its book of business to the pool by paying 75% of the gross premium that was charged to the insured. Then, in case an accident occurs, the PRR assumes all claim payments. The losses (gains) in the pool are then shared among the insurers in proportion of their market share.

Since automobile insurance is mandatory in Québec, the PRR was initially created in order to compensate insurers that were forced by the legislator to insure some risks that had been previously turned down by multiple insurers. The idea was that the insurer could then send these risks to the pool and the losses would be spread among all insurers operating in the province. Of course, such extreme risks represent far less than the allowed 10% of an insurer's volume. The difference can then be used for other profitable purposes. One possibility is for an insurer to ceed the risks that bring most volatility in the book of business and the pool therefore becomes a means of obtaining reinsurance. In this section, we take a different view: our interest is to use highly discriminative models to identify "holes" in the ratemaking model, i.e., to identify the risks that have been underpriced the most. Mathematically, this correspond to identifying risks for which the expected value

of the claims is higher than 75% of the gross premium, i.e., those risks with an expected loss ratio of at least 75%, a figure above the industry's average performance. For a particular insurer, the lower the loss ratio, the more difficult it becomes to identify risks that can be (statistically) profitably ceeded. Still, there are a few reasons why important underpricings can be identified:

1. legislation related to ratemaking is more restrictive than the one that pertains to the risk-sharing pool,

2. strategic marketing concerns may have forced the insurer to underprice a certain part of its book of business and,

3. other concerns may not allow the insurer to use highly discriminative models for the purpose of ratemaking.

The last two items can possibly be handled by rule-based systems if the insurer clearly knows which segments of its book of business are underpriced. The legislative context is of more interest to us: stringent legislators refrain insurers from using highly explanatory variables such as sex or age for the purpose of ratemaking. If the pool facility ruling is silent in that regard, then underpricings can easily be identified. But this can be done with traditional models.

The interest in highly discriminative models such as Neural Networks comes from the necessity of filing ratemaking plans in a clear fashion. Often, this filing operation limits an actuary in his desire to exploit relevant dependencies between explanatory variables. A lot of insurers still analyze variables independently, in *silos*, in order to compute individual parameters for each one of them. In that case, no dependency can be captured unless a "home-brewed" variable, resulting from the combination of many, is added. But this is a highly informal procedure which relies on the actuary's thorough knowledge of the problem at hand and technical assistance such as visualization tools. Neural Networks are able to automate this procedure and capture the most relevant of these dependencies w.r.t ratemaking. This is where comes in the most important difference between Neural Networks and Generalized Linear Models: automating the detection of depencies.

The superiority of the Neural Network Model is illustrated in Figures 9.10 and 9.11, where we have simulated the profits that can be generated by an insurer with 100M$ book of business operating at a global loss ratio of 65%. We compare Neural Networks and Generalized Linear Models as they take

turns as ratemaking model and facility model (the model used to identify underpricings in the ratemaking and to choose the risks to be ceeded to the facility). We measured profits as such: for a particular insured risk, let P_r and P_f be the premiums computed according to the ratemaking and facility models, respectively. Let C be the level of claims that occured for that risk (usually zero). The premiums P_r and P_f are pure premiums. Since we have assumed a loss ratio of 65%, we can compute the gross premium as $P_r/65\%$ for the ratemaking model. Then, when a risk is ceeded, the facility keeps 75% of that premium. So the actual profit of ceeding a particular risk is

$$\text{Actual Profit} \;=\; C \;-\; 75\% \;\times\; P_r/65\%.$$

Similarly, the facility premium P_f corresponds to the expected level of claims, so the projected profit of ceeding a risk is:

$$\text{Projected Profit} \;=\; P_f \;-\; 75\% \;\times\; P_r/65\%.$$

Accordingly, the facility premium must be 15.4% higher than the corresponding ratemaking premium in order to profitably (statistically) ceed a risk. Figure 9.10 shows that the Neural Network, used as a facility model can help generate substantial profits (between 1.25M$ and 1.5M$ for the 100M$ book of business insurer) when a GLM is used for ratemaking. It profitably identifies underpricings on more than 10% of the insurer's book of business. Also observe that the difference between the actual and relative profits is relatively small. Since

$$\text{Actual Profit - Projected Profit} \;=\; C \;-\; P_f,$$

we conclude that the Neural Network is very precise at estimating the expected claims level for high risks.

According to the graphic, the insurer has been able to ceed

$$1.25\text{M\$} \;+\; 75\% \times\; 10\% \;\times\; 100\text{M\$} \;=\; 8.75\text{M\$}$$

in claims to the pool. Thus, the ceeded risks had an average loss ratio of 87.5% up from the global figure of 65%.

On the other hand, Figure 9.11 shows that the GLM model, when used as the facility model mistakingly identifies underpricings in the ratemaking Neural Network model that appear in the projected profit but do not translate in real, actual profit.

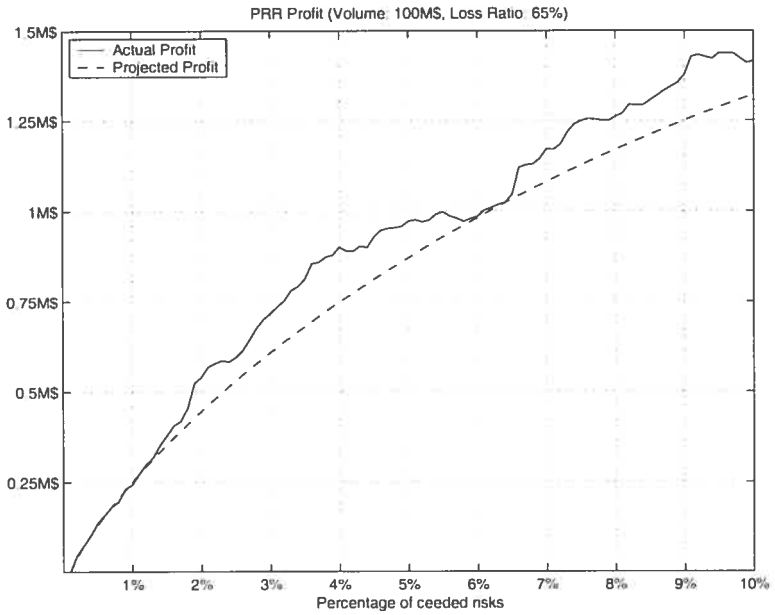

Figure 9.10: Profit from the PRR facility as a function of the ceeding percentage. Both, the projected profit (dashed) and the actual profit (solid) are shown. These illustrations apply to an insurer with a volume of business of 100M$ and a global loss ratio of 65%. The benchmark model is the GLM model and the model used to identify the underpricings is the Neural Network model. This leads to an actual profit of over 1.25M$ in savings.

9.8 Conclusion

Neural networks have been known to perform well in tasks where discrimination is an important aspect of the task at hand and this has lead to many commercially successful application of these modeling tools (62). We have shown that, when applied properly while taking into account the particulars of insurance data, that ability to discriminate is also revealed with insurance data. When applied to automobile insurance ratemaking, they allow us to identify more precisely the true risk associated to each insured.

We have argued in favor of the use of statistical learning algorithms such as Neural networks for automobile insurance ratemaking. We have described various candidate models and compared them qualitatively and numerically.

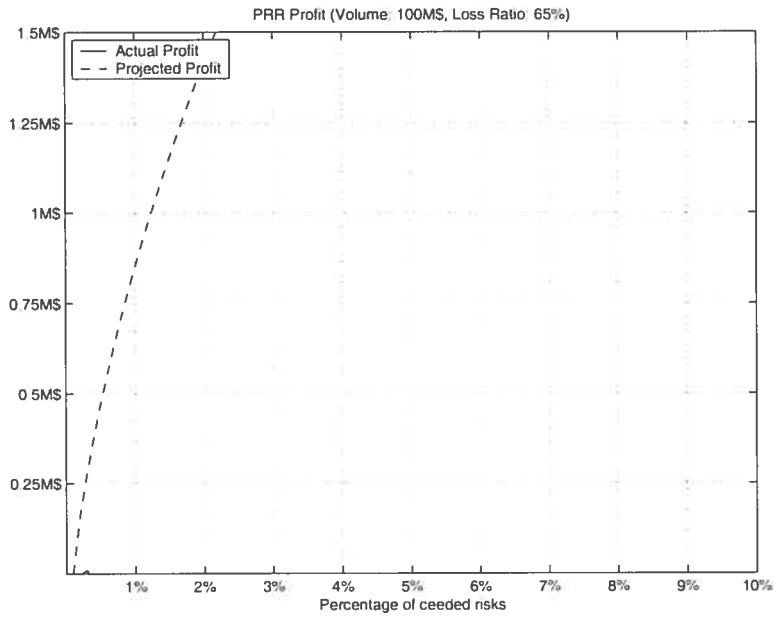

Figure 9.11: Profit from the PRR facility as a function of the ceeding percentage. Both, the projected profit (dashed) and the actual profit (solid) are shown. These illustrations apply to an insurer with a volume of business of 100M$ and a global loss ratio of 65%. The benchmark model is the Neural Network model and the model used to identify the underpricings is the GLM model. Top: the Neural Network model the GLM model projects very large profits that do not materialize on test data.

We have found that the selected model has significantly outperformed all other models, including the current premium structure. We believe that their superior performance is mainly due to their ability to capture high-order dependencies between variables and to cope with the fat tail distribution of the claims. Other industries have adopted statistical learning algorithms in the last decade and we have shown them to be suited for the automobile insurance industry as well.

Completely changing the rate structure of an insurer can be a costly enterprise, in particular when it involves significant changes in the computer systems handling transactions, or the relations with brokers. We have shown that substantial profit can be obtained from the use of Neural Networks in

the context of risk-sharing pools. There are still many other applications where better discrimination of risks can be used profitably, in particular target marketing, fraud detection and elasticity modelling.

Target Marketing:
When an insurer sends out mail solicitation, only a portion (5%-10%) of the whole population will be contacted. The goal here is, given this fixed portion, to reach the maximum number of people who will respond positively to the solicitation. Another possibiity would be for the insurer to develop a "customer lifetime value" model that would predict, given an insured's profile, what is the expected present value of the future profits that will be generated by acquiring this particular insured's business. Then, by using the customer lifetime value model in conjunction with a model for the probability of positive response, an insurer could attempt to maximize the profit of its solicitation campaign instead of simply maximizing the number of new insureds.

Fraud Detection
Fraud represents 10%-15% of all claims. Usually only a portion of the claims will be looked at by an insurer's investigators. The goal of fraud detection is to develop a model that will help an insurer increase the effectiveness of its investigators by referring them cases that are more likely to be fraudulent. In order to do so, one needs a database of previous successful and unsuccessful investigations. Neural Networks have been applied with great success to credit card fraud detection.

Elasticity Modelling
The greatest benefit from an improved estimation of pure premium derives by considering its application to ratemaking. The main reason for these benefits is that a more discriminant predictor will identify a group of insureds that are significantly undercharged and a (much larger) group that is significantly overcharged. Identifying the *undercharged* will **increase profits**: increasing their premiums will either directly increase revenues (if they stay) or reduce underwriting losses (if they switch to another insurer). The advantage of identifying the insured profiles which correspond to *overcharged* premiums can be coupled with a marketing strategy in order attract new customers and **increase market share**, a very powerful engine for increased profitability of the insurer (because of the fixed costs being shared by a larger number of

insureds).

To decide on the appropriate change in premium, one also needs to consider market effects. An elasticity model can be independently developped in order to characterize the relation between premium change and the probability of loosing current customers or acquiring new customers. A pure premium model such as the one described in this chapter can then be combined with the elasticity model, as well as pricing constraints (e.g. to prevent too much rate dislocation in premiums, or to satisfy some jurisdiction's regulations), in order to obtain a function that "optimally" chooses for each insured profile an appropriate change in gross premium, in order to maximize a financial criterion.

Clearly, the insurance industry is filled with analytical challenges where better discrimination between good and bad risks can be used profitably. We hope this chapter goes a long way in convincing actuaries to include Neural networks within their set of modeling tools for ratemaking and other analytical tasks.

9.9 Proof of the equivalence of the fairness and precision criterions

In this section, we show that, when all subpopulations are considered to evaluate fairness, the precision criterion and the fairness criterion, as they were defined in section 9.3, both yield the same premium function.

Theorem 9.9.1 *The premium function which maximises precision (in the sense of equation 9.2) also maximises fairness (in the sense of equation 9.5, when all subpopulations are considered), and it is the only one that does maximize it.*

Proof:

Let P be a subset of the domain of input profiles. Let q be a premium predictor function. The bias in P is defined by

$$b_q(P) = \frac{1}{|P|} \sum_{(x_i,a_i)\in P} (q(x_i) - a_i).$$

Let $F_q = -E[\sum_P b_q(P)^2]$ be the expected "fairness" criterion using premium function q, to be maximized (by choosing q appropriately).

Let $p(x) = E[a|x]$ be the optimal solution to the precision criterion, i.e. the minimizer of

$$E[(p(X) - A)^2].$$

Consider a particular population P. Let $q(P)$ denote the average premium for that population using the premium function $q(x)$,

$$q(P) = \frac{1}{|P|} \sum_{(x_i,a_i)\in P} q(x_i)$$

and similarly, define $a(P)$ the average claim amount for that population,

$$a(P) = \frac{1}{|P|} \sum_{(x_i,a_i)\in P} a_i$$

Then the expected squared bias for that population, using the premium function q, is

$$E[b_q(P)^2] = E[(q(P) - a(P))^2]$$

which is minimized for any q such that $q(P) = E[a(P)]$.

Note in particular that the optimal ESE solution, p, is such a minimizer of F_q, since

$$p(P) = \frac{1}{|P|} \sum_{(x_i,a_i)\in P} E[a_i|x_i] = E[\frac{1}{|P|} \sum_{(x_i,a_i)\in P} a_i] = E[a(P)]$$

We know therefore that $q = p$ is a minimizer of F_q, i.e. $\forall q, F_p \leq F_q$.

Are there other minimizers? Consider a function $q \neq p$, that is a minimizer for a particular population P_1. Since $q \neq p$, $\exists x$ s.t. $q(x) \neq p(x)$. Consider the particular singleton population $P_x = \{x\}$. On singleton populations, the expected squared bias is the same as the expected squared error. In fact, there is a component of F which contains only the squared biases for the singleton populuations, and it is equal to the expected squared error. Therefore on that population (and any other singleton population for which $q \neq p$) there is only one minimizer of the expected squared bias, and it is the conditional expectation $p(x)$. So $E[(q(x) - A)^2|X = x] > E[(p(x) - A)^2|X = x]$ and therefore $E[b_q(P_x)] > E[b_p(P_x)]$. Since p is a maximiser of fairness for all populations, it is enough to prove that q is sub-optimal on one population to prove that the overall fairness of q is less than that of p, which is the main statement of our theorem:

$$\forall q \neq p, F_q > F_p.$$

□

Chapitre 10

Synthèse

Dans le chapitre précédent, nous avons montré que lorsque les réseaux de neurones sont appliqués en tenant compte des particularités de l'assurance, on peut exploiter leur pouvoir discriminant pour mieux mesurer le risque réel que représente un assuré.

Nous avons argumenté en faveur de l'utilisation, pour l'assurance automobile, de modèles tels les réseaux de neurones et de méthodologies de sélection de modèle telles que celles qui sont couramment utilisées dans le milieu des algorithmes d'apprentissage. Nous avons décrit plusieurs types de modèles et les avons comparés de façon qualitative et quantitative. Nous avons trouvé que le modèle choisi et proposé avait significativement battu les autres modèles, incluant le modèle actuellement utilisé pour la tarification. Nous croyons que cette supériorité dans la performance est due à la capacité qu'ont les réseaux de neurones de capter les interdépendances de haut niveau entre les facteurs explicatifs et aussi à l'utilisation d'une architecture qui permet d'accroître la robustesse des algorithmes utilisés aux distributions asymmétriques et à queues longues et épaisses. Nous en tirons la conclusion que l'assurance automobile, tout comme plusieurs autres industries pourraient utiliser à profit des modèles tels les réseaux de neurones.

Nous avons identifié un problème analytique, connexe à la tarification, appelé plan de répartition des risques. La particularité de ce problème est que la modification des modèles utilisés n'affecte pas les assurés, ni les courtiers. Afin de bien performer à cette tâche, il suffit d'utiliser le meilleur modèle possible. Nous avons décrit d'autres applications potentielles dans le milieu de l'assurance automobile : marketing ciblé, détection de fraudes et modélisation de l'élasticité dans le marché.

Chapitre 11

Conclusion

Dans ce chapitre de conclusion, nous positionons les résultats obtenus dans le contexte des différentes applications potentielles des algorithmes d'apprentissage aux risques financiers. Ceci nous permet d'établir des liens entre les trois parties de cette thèse qui peuvent sembler disparates. Des expériences réalisées par l'auteur au cours des derniéres années découlent non seulement les résultats obtenus, publiés et reproduits dans cette thèse mais aussi une certaine intuition quant à la pertinence, en termes de profitabilité, pouvant découler de l'implantation d'algorithmes performants pour la résolution des problèmes analytiques financiers.

Les problèmes analytiques financiers baignent souvent dans un contexte qui inclut des considérations d'ordre légal, de mise en marché, etc. De façon générale, une personne qui cherche à valoriser son travail lié au développement de nouveaux algorithmes d'apprentissage devrait tenter d'identifier les applications pour lesquelles l'aspect analytique de la solution prévaut par rapport à ces autres considérations. À cet effet, un élément essentiel à considérer est la standardisation du produit. Chaque fournisseur d'un produit tente d'attirer une clientèle toujours plus nombreuse et pour se faire, doit se distinguer de ses concurrents. Plus le produit est standard d'un fournisseur à l'autre, plus l'importance du prix, comme élément distinctif, croît. Dans ce contexte, les efforts analytiques pour l'obtention d'un prix optimal tenant compte de la valeur économique du client, sont valorisés. Forts de ce mode d'analyse, nous pouvons maintenant nous plonger dans les divers défis analytiques financiers.

11.1 Assurances

Le milieu des assurances comprend deux types importants de produits : les assurances de personnes, aussi appelées assurances-vie et les assurances de biens ou assurances générales. Cette distinction est à ce point importante que plusieurs compagnies n'opèrent que dans l'un des deux types d'assurance. La formation actuarielle elle aussi se scinde en ces deux branches. Nous décrivons ici les enjeux propres à chacun de ces deux types d'assurance. Ensuite, nous abordons les différents éléments analytiques importants de cette industrie et communs aux deux types d'assurance.

11.1.1 Assurances-vie

La problématique actuarielle dans le mileu de l'assurance-vie se caractérise par des projections financières à long terme puisque les primes actuellement versées serviront au paiement de prestations versées jusquà 100 ans plus tard. Le nombre de facteurs explicatifs généralement utilisés est relativement faible. Traditionnellement, les primes d'assurances étaient établies en ne tenant compte que de trois facteurs : âge, sexe et statut fumeur ou non-fumeur. Aujourd'hui, le nombre de facteurs utilisés par les actuaires et pouvant entrer en jeu dans la tarification tend à croître et les algorithmes d'apprentissage deviendront probablement de plus en plus importants pour ce milieu. En particulier, l'utilisation potentielle de toute l'information génétique provenant de l'ADN peut faire rêver les adeptes des algorithmes d'apprentissage. Il serait alors possible d'évaluer la propension d'une personne à développer certaines maladies.

Bien que le but ultime d'une compagnie d'assurance soit de maximiser la profitabilité de ses opérations, la problématique est généralement formulée sous forme d'estimation de densité, appelée *graduation des tables* et consiste à estimer le taux de mortalité d'un assuré aujourd'hui et de projeter ce taux dans le futur. Utilisant les données empiriques disponibles, les actuaires tentent de balancer, par l'utilisation de méthodes diverses, *la précision et le lissage*, ce qui correspond essentiellement au dilemme biais-variance.

Un élément essentiel du travail actuariel est la projection de l'évolution générale des taux de mortalité. Il sagit de quantifier l'impact futur des efforts actuels en recherche médicale et de l'éclosion potentielle de maladies importantes telles SIDA et SRAS. Ces projections peuvent avoir un impact important mais ne se prêtent pas facilement á l'évaluation rigoureuse par

sélection de modèle, étant donné la non-stationarité de l'environnement et le long délai entre la tarification sur la base de ces projections et l'observation des données qui s'y rapportent.

En plus de raffiner l'évalution des risques de mortalité, les compagnies peuvent aussi tenter de *couvrir* ("hedge") ces risques. Notons qu'un assureur s'expose à un risque de mortalité par l'émission d'une assurance-vie et un risque de survie par l'émission d'une rente viagère. Par un balancement judicieux de ces deux types de produits, un assureur peut donc réduire substantiellement l'importance des risques auxquels il s'expose. De même, un assureur peut se prémunir contre les risques de rendement des marchés financiers par l'utilisation de véhicules financiers permettant l'appariement des flux financiers découlant de ses passifs ("asset-liability management").

Enfin, les produits d'assurance-vie sont généralement complexes (p.ex. assurance-vie universelle) et, à l'exception de produits tels l'assurance-vie temporaire 10 ans ou 100 ans (aussi appelée assurance-vie entière), peu standardisés. La tendance à englober les produits d'assurance dans une stratégie financière globale incluant les régimes enregistrés d'épargne retraite, hypothèques, investissements dans les marchés financiers rend complexe la comparaison entre les produits de différents fournisseurs. Ces possibilités de différentiation autres que par le biais du prix des produits réduit l'importance de la précision dans la mesure des taux de mortalité.

En conclusion, la possibilité de tenir compte des informations tirées de l'ADN semble être l'avenue la plus intéressante et où l'utilisation d'algorithmes d'apprentissage pourrait s'avérer la plus profitable. Bien sûr, cette possibilité soulève des questions éthiques. L'adoption de lois contrôlant l'utilisation de ces informations ne sera pas uniforme et mènera probablement à des défis analytiques qui diffèreront d'une juridiction à l'autre.

11.1.2 Assurances générales

Dans le domaine des assurances générales, environ la moitié des revenus proviennent de l'assurance automobile, le tiers de l'assurance habitation et le reste d'autres types d'assurance moins importants (en terme de volume) dont la responsabilité civile, les récoltes, etc. Dans cette sous-section, nous nous attardons à l'assurance automobile.

L'assurance automobile est probablement le produit le plus standardisé de toute l'industrie de l'assurance. Partout en Amérique du Nord, l'assurance automobile est obligatoire et les couvertures offertes sont réglementées et

bien définies. Il devient donc difficile pour un assureur de distinguer son produit de ceux offerts par ses concurrents autrement que par le biais du prix. Les contrats d'assurance automobile sont généralement d'une durée de un an donc beaucoup plus courts que les contrats d'assurance-vie ou de rentes viagères. Ainsi, au bout d'un an, on peut confronter un modèle avec les résultats obtenus et les méthodes de sélection de modèle peuvent jouer un rôle important.

Le nombre de facteurs explicatifs, appelés critères de tarification, utilisés par les assureurs peut aller jusqu'à 50. La malédiction de la dimensionalité est donc bien présente et les actuaires en sont conscients. Clairement, les algorithmes d'apprentissage et les méthodes de sélection de modèle pourraient jouer un rôle important dans le milieu de l'assurance automobile au cours de prochaines années. Le nombre de facteurs pourrait augmenter de façon dramatique si l'utilisation de boîtes noires devenait répandue. Cette possibilité semble encore toutefois lointaine, certainement plus que celle liée à l'utilisation des informations génétiques en assurance-vie.

Le problème de la modélisation en assurance automobile est généralement scindé en deux : fréquence et sévérité. La fréquence représente le nombre de réclamations dans une période donnée, généralement un an. La sévérité représente le montant d'un sinistre. Puisque la plupart des assurés ne réclament pas au cours d'une année donnée, alors l'espérance conditionnelle de la fréquence doit être estimée à l'aide d'ensembles fortement débalancés. Par exemple, en assurance automobile, le taux de sinistre est d'environ 15%. Si les blessures corporelles et la responsabilité civile sont couvertes par l'assureur et que les assureurs peuvent se poursuivre entre eux, alors la sévérité d'une réclamation peut atteindre des millions de dollars. La distribution de la sévérité exhibe donc une aile lourde du côté des montants positifs. Pour cette raison, au chapitre 9, nous avons présenté des modèles de mixtures d'experts qui améliorent la robustesse de l'apprentissage par rapport aux très grandes réclamations.

Analysons les barrières à franchir avant d'en arriver, par exemple, à l'application des réseaux de neurones pour la tarification en assurance automobile. Premièrement, la disponibilité des données n'est pas acquise. Au cours des années 90, plusieurs assureurs ont entrepris des projets de développement d'entrepôts de données ("data warehouses"). Dans plusieurs cas, ces projets n'ont pas abouti ce qui rend plus difficile toute tâche de forage de données ("data mining").

Dans certaines provinces et états américains, le législateur ne permet pas

l'utilisation de technologies complexes pour la discrimination entre bons et mauvais risques. Le but est souvent de protéger l'equité actuarielle ("actuarial fairness"). Aux fins de vérification de cette équité, le législateur exige parfois même de pouvoir connaître les valeurs des paramètres associés à certains facteurs tels que le sexe ou le territoire. Dans certaines juridictions, un important travail de vulgarisation devra être réalisé auprès des autorités avant que les assureurs puissent baser leurs tarifs sur des modèles tels que les réseaux de neurones. Notons sur ce point que la législation du Québec est généralement reconnue comme étant plus libérale.

Les courtiers et agents aiment pouvoir expliquer et reproduire le calcul d'une prime d'assurance. Les actuaires eux-mêmes préfèrent utiliser des modèles dont les paramètres leurs sont familiers et pour lesquels ils ont développé une certaine intuition. Les réseaux de neurones sont souvent perçus comme étant des boîtes noires. On comprend ici l'ampleur du travail à réaliser avant d'en venir à une acceptation de modèles tels les réseaux de neurones.

En conclusion, la problématique de l'assurance automobile se prête fort bien aux algorithmes d'apprentissage et techniques de sélection de modèle. Les barrières restantes sont essentiellement culturelles. Toutefois, plutôt que les attaquer de front, on peut les contourner en appliquant les algorithmes d'apprentissage aux autres éléments analytiques auxquels font face les assureurs, leur permettant ainsi de se familiariser avec les algorithmes plus sophistiqués proposés. La section 9.7 traite de l'application des réseaux de neurones au Plan de Répartition des Risques, sans doute l'application la plus simple, et appropriée de toute l'industrie des assurances puisque sans effet sur les parts de marché d'un assureur et essentiellement libre de contraintes législatives. Enfin, à la section 9.8, les problématiques de marketing ciblé, modélisation de l'élasticité et détection de fraudes sont décrites.

11.2 Marchés Financiers

L'application des algorithmes d'apprentissage aux marchés financiers est beaucoup mieux connue, documentée et a attiré beaucoup plus d'attention de la part des chercheurs, principalement au cours de la dernière décennie. La possibilité de pouvoir prédire les rendements des titres financiers est certainement très séduisante. La profitabilité de toute amélioration du pouvoir prédictif des modèles peut mener à l'obtention de gains faramineux, en particulier lorsque conjuguée avec l'utilisation de produits dérivés. Nous brossons

ici un rapide portrait des principaux problèmes analytiques des marchés financiers et d'intérêt pour les spécialistes des algorithmes d'apprentissage.

Au chapitre 3, nous nous sommes attardés à la prédiction des rendements et de la volatilité d'actifs financiers et au chapitre 6, à la valorisation d'options financières. Ces problématiques ont été décrites dans les chapitres correspondants. D'autres éléments analytiques importants sont d'intérêt pour les chercheurs en apprentissage statistique, nous en abordons deux.

Tout d'abord, un investisseur dispose généralement d'un montant fixe qu'il désire investir. Il reste à cet investisseur de choisir les véhicules financiers qui lui conviennent. Un investisseur pourra vouloir maximiser l'espérance du rendement obtenu, ou maximiser un rendement normalisé par une mesure de risque associée à la stratégie retenue, tel qu'avec le Sharpe Ratio, ou enfin maximiser l'espérance de l'utilité qu'il tirera de sa richesse future. C'est là ce que l'on appelle l'allocation d'actif.

Les observations empiriques tendent à montrer que les rendements des actifs financiers ne sont pas gaussiens : la distribution de ces rendements est caractérisée par la présence d'ailes lourdes. Aussi, la probabilité d'événements extrêmes suscite beaucoup d'attention, en voici deux exemples.

Selon le calcul de la VaR ("Value at Risk") on s'intéresse au montant qui pourrait être perdu au cours d'une période donnée à venir, cette période pouvant aller d'une journée à une année. Le cas trivial d'une perte totale ne nous apporte aucune information et donc on calcule plutôt la différence entre la valeur courante du portefeuille considér'e et la borne inférieure d'un intervalle de confiance sur la distribution de la valeur de ce portefeuille au terme de la période d'intérêt. Cette différence, fortement influencée par le choix du pourcentage de l'intervalle de confiance, est la valeur à risque. Des mesures plus raffinées doivent tenir compte des interdépendances entre les rendements des différents actifs qui composent un portefeuille. La valeur à risque peut aussi être calculée de façon conditionnelle à un ensemble de facteurs. La valeur à risque n'est pas une fin en soi mais plutôt une exigence légale. Un raffinement dans le calcul de cette valeur peut apporter un gain si elle permet à l'institution de gérer des options de vente : on voudra acheter des options de vente lorsque la valeur à risque devient trop élevée.

En actuariat, une part importante de la recherche universitaire est consacrée à l'estimation de la probabilité de ruine. Étant donné un certain niveau de capital initial, on veut estimer la probabilité qu'une compagnie d'assurance tombe en faillite au cours d'une période donnée à venir. Ce problème est très similaire au calcul de la valeur à risque : dans le cas précédent, on fixe

la probabilité et on estime la valeur qui y correspond. Dans le cas de la probabilité de ruine, la valeur de la perte est fixée et c'est la probabilité qu'on désire estimer.

La modélisation en finance est souvent réalisée dans un cadre mathématique formel et le nombre de facteurs explicatifs utilisés est limité, Les outils d'apprentissage statistique permettent de tenir compte d'une richesse d'information beaucoup plus grande et pour cette raison, ils devraient connaître beaucoup de succès dans leur application aux divers risques financiers.

11.3 Banques

Dans le milieu bancaire, le calcul de cotes de crédit est fortement répandu. Le but est de mesurer le risque qu'un emprunteur ne puisse effectuer les paiements dûs en vertu de son emprunt. Les banques émettrices sont aussi responsables des fraudes commises sur les cartes de crédit qui sont leurs. La détection de fraudes est donc, pour ces banques, un problème important. Finalement, les banques consacrent beaucoup d'effort à l'estimation de la valeur économique du client ("customer lifetime value"). Cette valeur sert à déterminer l'ampleur des efforts qui seront déployés afin de conserver et développer la relation avec chacun de leur clients ou prospects.

Les réseaux de neurones et autres modèles complexes sont régulièrement utilisés pour la résolution de ces problèmes analytiques dans le milieu bancaire. Pour les chercheurs du milieu de l'apprentissage statistique, étant donné l'ampleur des sommes en jeu et l'ouverture aux méthodes proposées en apprentissage statistique, les défis analytiques du milieu bancaire représentent un excellent véhicule leur permettant de mettre en valeur les fruits de leur recherche.

11.4 Considérations particulières

Les données financières possèdent une structure dans le temps. Pour cette raison, des techniques telles que la validation croisée d'ordre K ("K-fold cross-validation") ne peut être appliquée puisque les données incluent théoriquement toute l'information du passé, incluant les prix passés si l'on se fie à la l'hypothèse faible de l'efficacité des marchés financiers.

Cette structure temporelle soulève une série de questions. Premièrement,

quelles sont les données du passé qui sont encore pertinentes pour la prédiction faite aujourd'hui des rendements futurs ? Cette question a été abordée au chapitre 3. Aux chapitres 6 et 9 cette question a été laissée de côté, au profit d'autres considérations : l'utilisation sur des données réelles d'une nouvelle architecture de réseaux de neurones et la modélisation en hautes dimensions en présence de bruit asymétrique et d'une aile lourde, respectivement. Bien que dans la pratique, le choix de la fenêtre de temps dont on extraiera les données pour l'entraînement des modèles soit importante, il n'est pas clair que cette fenêtre soit la même pour tous les modèles. Pour des fins de comparaisons et de sélection de modèles, il était plus simple de fixer la longueur de la fenêtre et de comparer les modèles étant donné cette fenêtre fixe que de choisir une longueur de fenêtre pour chacun des modèles utilisés.

Ensuite, pendant combien de temps un modèle conserve-t-il son pouvoir prédictif ? Autrement dit, quelle est la persistence du modèle ? Au chapitre 6, nous mesurons la persistence de quelques modèles et tirons des conclusions à ce sujet. Est-ce que la persistence peut dépendre de la fenêtre de temps qui a été utilisée pour l'entraînement des modèles ? Cette question reste en suspens. Dans le milieu de l'assurance, les actuaires réévaluent et apportent des correctifs à leurs modèles sur une base annuelle. La justesse de cette pratique pourra être validée dans le cadre de recherches futures.

La fréquente présence d'ailes lourdes dans la distribution des données complique la tâche de modélisation des données financières. En particulier, dans le domaine de l'assurance, ce problème nous a mené au développement d'un type différent de mixture d'experts. Une possibilité serait de transformer les valeurs cibles du processus par exemple en prenant le logarithme de ces valeurs. Toutefois, puisque selon l'inégalité de Jensen on a que $E[\log x] \leq log(E[x])$, alors la sortie du modèle devrait être ajustée par un autre modèle entraîné à cette fin.

11.5 Conclusion

Ce chapitre visait à placer les résultats de recherche de cette thèse dans le contexte plus large des problèmes liés à l'évaluation des risques financiers. Ce milieu est vaste et faste et les algorithmes d'apprentissage y seront certainement d'une grande utilité au cours des prochaines années, en particulier dans le domaine des assurances auquel les chercheurs ont peu prêté attention, où les défis analytiques sont nombreux et où les bases de données se prêtant à

l'analyse statistique sont maintenant disponibles ou sur le point de l'être.

Références

[1] H. Akaike. A new look at the statistical model identification. *IEEE Transations on Automatic Control*, AC-19(6) :716–728, 1974.

[2] J. Aldrich. R.A. Fisher and the making of maximum likelihood 1912-22. Technical Report 9504, University of Southampton, Department of Economics, 1995.

[3] D. Allen. The relationship between valriable selection and data augmentation and a method of prediction. *Technometrics*, 16 :125–127, 1977.

[4] R.A. Bailey and L. Simon. Two studies in automobile insurance ratemaking. *ASTIN Bulletin*, 1(4) :192–217, 1960.

[5] A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3) :930–945, 1993.

[6] R.E. Bellman. *Dynamic Programming*. Princeton University Press, NJ, 1957.

[7] Y. Bengio. Continuous optimization of hyper-parameters. Technical Report 1144, Département d'informatique et recherche opérationnelle, Université de Montréal, 1999.

[8] Y. Bengio and C. Dugas. Learning simple non-stationarities with hyper-parameters. Technical Report 1145, Département d'informatique et recherche opérationnelle, Université de Montréal, 1999.

[9] Y. Bengio and P. Frasconi. Input/Output HMMs for sequence processing. *IEEE Transactions on Neural Networks*, 7(5) :1231–1249, 1996.

[10] Y. Bengio and F. Gingras. Recurrent neural networks for missing or asynchronous data. In M. Mozer, D.S. Touretzky, and M. Perrone, editors, *Advances in Neural Information Processing System*, volume 8, pages 395–401. MIT Press, Cambridge, MA, 1996.

[11] F. Black and M. Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3) :637–654, 1973.

[12] T. Bollerslev. Generalized autoregressive conditional heteroskedastic process. *Journal of Econometrics*, 31 :307–327, 1986.

[13] L. Breiman. Bagging predictors. *Machine Learning*, 26(2) :123–140, 1996.

[14] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth Int. Group, 1984.

[15] R.L. Brown. Minimum bias with generalized linear models. In *Proceedings of the Casualty Actuarial Society*, 1988.

[16] J.Y. Campbell, A. W. Lo, and A.C. MacKinlay. *The Econometrics of Financial Markets*. Princeton University Press, Princeton, 1997.

[17] F. Cantelli. Sulla probabilita come limita della frequenza. *Rend. Accad. Lincei*, 26(1) :39, 1933.

[18] N. Chapados and Y. Bengio. Extensions to metric-based model selection. *Journal of Machine Learning Research*, 3 :1209–1227, March 2003. Special Issue on Feature Selection.

[19] N. Chapados, Y. Bengio, P. Vincent, J. Ghosn, C. Dugas, I. Takeuchi, and L. Meng. Estimating car insurance premia : a case study in high-dimensional data inference. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

[20] H. Chernoff. A measure of asymptotic efficiency of tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23 :493–507, 1952.

[21] P. Craven and G. Wahba. Smoothing noisy data with spline functions. *Numerical Mathematics*, 31 :377–403, 1979.

[22] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge Press, 2000.

[23] G. Cybenko. Continuous valued neural networks with two hidden layers are sufficient. Technical report, Department of Computer Science, Tufts University, Medford, MA, 1988.

[24] G. Cybenko. Approximation by superpositions of a sigmoidal function. 2 :303–314, 1989.

[25] M.C. Delfour and J.-P. Zolésio. *Shapes and Geometries : Analysis, Differential Calculus, and Optimization*. SIAM, 2001.

[26] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society B*, 39 :1–38, 1977.

[27] F. X. Diebold and R. S. Mariano. Comparing predictive accuracy. *Journal of Business and Economic Statistics*, 13(3) :253–263, 1995.

[28] C. Dugas, O. Bardou, and Y. Bengio. Analyses empiriques sur des transactions d'options. Technical Report 1176, Départment d'informatique et de Recherche Opérationnelle, Université de Montréal, Montréal, Québec, Canada, 2000.

[29] C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, and R. Garcia. A universal approximator of convex functions applied to option pricing. In *Advances in Neural Information Processing Systems*, volume 13, Denver, CO, 2001.

[30] C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, and R. Garcia. Incorporating second-order functional knowledge for better option pricing. *Journal of Machine Learning Research*, 2003. accepted.

[31] C. Dugas, Y. Bengio, N. Chapados, P. Vincent, G. Denoncourt, and C. Fournier. *Intelligent Techniques for the Insurance Industry*, chapter Statistical Learning Algorithms Applied to Automobile Insurance Ratemaking. World Scientific, 2003.

[32] C. Dugas, Y. Bengio, N. Chapados, P. Vincent, G. Denoncourt, and C. Fournier. Statistical learning algorithms applied to automobile insurance ratemaking. *CAS Forum*, 2003.

[33] William Evans, Sridhar Rajagopalan, and Umesh Vazirani. Choosing a reliable hypothesis. In *Proceedings of the 6th Annual Conference on Computational Learning Theory*, pages 269–276, Santa Cruz, CA, USA, July 1993. ACM Press.

[34] R. A. Fisher. On an absolute citerion for frequency curves. *Messenger of Mathematics*, 41 :155–160, 1912.

[35] R. A. Fisher. Frequency distribution of the values of the correlation coefficient in samples from an indefinitely large population. *Biometrika*, 10 :507–521, 1915.

[36] R. A. Fisher. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London*, A222 :309–368, 1922.

[37] R. A. Fisher. Theory of statistical estimation. *Proceedings of the Cambridge Philosophical Society*, 22 :700–725, 1925.

[38] R. Garcia and R. Gençay. Pricing and Hedging Derivative Securities with Neural Networks and a Homogeneity Hint. Technical Report 98s-35, CIRANO, Montréal, Québec, Canada, 1998.

[39] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1) :1–58, 1992.

[40] Z. Ghahramani and M. I. Jordan. Supervised learning from incomplete data via an EM approach. In J.D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, San Mateo, CA, 1994. Morgan Kaufmann.

[41] F. Gingras, Y. Bengio, and C. Nadeau. On out-of-sample statistics for time-series. In *Computational Finance 2000*, 2000.

[42] V. Glivenko. Sulla determinazione empirica delle leggi di probabilita. *Giornale dell'Istituta Italiano degli Attuari*, 4 :92, 1933.

[43] F.R. Hampel, E.M. Ronchetti, P.J. Rousseeuw, and W.A. Stahel. *Robust Statistics, The Approach based on Influence Functions*. John Wiley & Sons, 1986.

[44] B. Hassibi and D. G. Stork. Second order derivatives for network pruning : Optimal brain surgeon. pages 164–171, San Mateo, CA, 1993. Morgan Kaufmann.

[45] T. Hastie, R. Tibshirani, and J. Friedman. *Data Mining, Inference and Prediction*. Springer, 2001.

[46] D. Haussler, J. Kivinen, and M.K. Warmuth. Sequential prediction of individual sequences under general loss functions. In *Computational Learning Theory, 2nd European Conference, EuroCOLT'95*, pages 69–83. Springer, 1995.

[47] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Mathematical Association*, 58 :13–30, 1963.

[48] A.E. Hoerl and R. Kennard. Ridge regression : Biased estimation for nonorthogonal problems. *Technometrics*, 12 :55–67, 1970.

[49] K.D. Holler, D. Sommer, and G. Trahair. Something old, something new in classification ratemaking with a novel use of glms for credit insurance. *Casualty Actuarial Society Forum*, pages 31–84, 1999.

[50] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. 2 :359–366, 1989.

[51] P. J. Huber. Robust estimation of a location parameter. *Annals of Mathematical Statistics*, 35 :73–101, 1964.

[52] P. J. Huber. A robust version of the probability ratio. *Annals of Mathematical Statistics*, 36 :17–53, 1965.

[53] P. J. Huber. Strict efficiency excludes superefficiency. *Annals of Mathematical Statistics*, 37 :14–25, 1966.

[54] P. J. Huber. Robust confidence limits. *Zeitschrift für Wahrscheinlichkeits Theorie*, 10 :269–278, 1968.

[55] P. J. Huber. *Robust Statistics*. John Wiley & Sons, New York, 1981.

[56] P. J. Huber. Projection pursuit. *Annals of Statistics*, 13 :435–525, 1985.

[57] P. J. Huber and H. Rieder. *Robust statistics, data analysis, and computer intensive methods*. Number 109 in Lecture Notes in Statistics. Springer, New York, 1996.

[58] P.J. Huber. *Robust Statistics*. John Wiley & Sons Inc., 1982.

[59] J.M. Hutchinson, A.W. Lo, and T. Poggio. A nonparametric approach to pricing and hedging derivative securities via learning networks. *Journal of Finance*, 49(3) :851–889, 1994.

[60] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixture of local experts. *Neural Computation*, 3 :79–87, 1991.

[61] G.V. Kass. An exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, 29(2) :119–127, 1980.

[62] Paul E. Keller. Neural networks : Commercial applications, 1997. http ://www.emsl.pnl.gov :2080/proj/neuron/neural/products.

[63] A. Kolmogorov. Sulla determinazione empirica di una leggi di distribuzione. *Giornale dell'Istituta Italiano degli Attuari*, 4 :33, 1933.

[64] M. Leshno, V. Lin, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6 :861–867, 1993.

[65] N. Littlestone and M.K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2) :212–261, 1994.

[66] F.X. Diebold R.S. Mariano. Comparing predictive accuracy. *Journal of Business and Economic Statistics*, 13 :253–263, 1995.

[67] P. McCullagh and J. Nelder. *Generalized Linear Models*. Chapman and Hall, London, 1989.

[68] J. Moody. *Prediction Risk and Architecture Selection for Neural Networks*. Springer, 1994.

[69] K. Murphy, M.J. Brockman, and P.K.W. Lee. Using generalized linear models to build dynamic pricing systems. *Casualty Actuarial Society*

Forum, pages 107–139, 2000.

[70] Claude Nadeau and Yoshua Bengio. Inference for the generalization error. In S.A. Solla, T.K. Leen, and K-R. Mller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 307–313. MIT Press, 2000.

[71] J.A. Nelder and R.W.M. Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society*, 135 :370–384, 1972.

[72] W. Newey and K. West. A simple, positive semi-definite, heteroscedasticity and autocorrelation consistent covariance matrix. *Econometrica*, 55 :703–708, 1987.

[73] T. Poggio, V. Torre, and C. Koch. Computational vision and regularization theory. *Nature*, 317(26) :314–319, 1985.

[74] P.J. Rousseeuw and A.M. Leroy. *Robust Regression and Outlier Detection.* John Wiley & Sons Inc., 1987.

[75] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning representations by back-propagating errors. *Nature*, 323 :533–536, 1986.

[76] R.E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2) :197–227, 1990.

[77] B. Schölkopf and A. Smola. *Learning with Kernels.* MIT Press, 2001.

[78] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10 :1299–1319, 1998.

[79] M. Stone. Cross-validatory choice and assesment of statistical predictions. *Journal of the Royal Statistical Society*, 36 :111–147, 1974.

[80] M. Stone. An asymptotic equivalence of choice of model by cross-validation and akaike's criterion. *Journal of the Royal Statistical Society*, 39 :44–47, 1977.

[81] R. Sutton and A. Barto. *An Introdiction to Reinforcement Learining.* MIT Press, 1998.

[82] I. Takeuchi, Y. Bengio, and T. Kanamori. Robust regression with asymmetric heavy-tail noise distributions. *Neural Computation*, 2002. to appear.

[83] A.N. Tikhonov and V.Y. Arsenin. *Solutions of Ill-posed Problems.* W.H. Winston, Washington D.C., 1977.

[84] V. Vapnik. *Statistical Learning Theory.* John Wiley, Lecture Notes in Economics and Mathematical Systems, volume 454, 1998.

[85] V.N. Vapnik. *Estimation of Dependences Based on Empirical Data.* Springer-Verlag, Berlin, 1982.

[86] V.N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, New York, 1995.

[87] D.H. Wolpert. Stacked generalization. *Neural Networks*, 5(241–259), 1992.