2003s-20

Comment améliorer la capacité de généralisation des algorithmes d'apprentissage pour la prise de décisions financières

Nicolas Chapados, Yoshua Bengio

Série Scientifique
Scientific Series

Montréal
Mai 2003

CIRANO
Centre interuniversitaire de recherche en analyse des organisations

CIRANO

Le CIRANO est un organisme sans but lucratif constitué en vertu de la Loi des compagnies du Québec. Le financement de son infrastructure et de ses activités de recherche provient des cotisations de ses organisations-membres, d'une subvention d'infrastructure du ministère de la Recherche, de la Science et de la Technologie, de même que des subventions et mandats obtenus par ses équipes de recherche.

CIRANO is a private non-profit organization incorporated under the Québec Companies Act. Its infrastructure and research activities are funded through fees paid by member organizations, an infrastructure grant from the Ministère de la Recherche, de la Science et de la Technologie, and grants and research mandates obtained by its research teams.

Les organisations-partenaires / The Partner Organizations

ISSN 1198-8177

Comment améliorer la capacité de généralisation des algorithmes d'apprentissage pour la prise de décisions financières

Nicolas Chapados,[*] *Yoshua Bengio*[†]

Résumé / *Abstract*

Ce rapport présente et propose plusieurs méthodes pour améliorer la capacité de généralisation des algorithmes d'apprentissage dans un contexte de prise de décisions financières. Globalement, ces méthodes visent à contrôler la capacité des algorithmes d'apprentissage en vue de limiter le problème du sur-apprentissage, qui est l'un des plus pernicieux en finance à cause des niveaux de bruit élevés rencontrés en pratique. Nous proposons quelques pistes de recherches afin d'améliorer les algorithmes et résultats déjà obtenus.

Mots clés : Performance de généralisation, méthodes métriques, sélection de modèles, régularisation, programmation dynamique approximative, apprentissage par renforcement.

This report presents and proposes several methods to improve the capacity of generalization of the learning algorithms in a context of financial decision-making. These methods, overall, aim at controlling the capacity of the learning algorithms in order to limit the problem of the over-training, which is one of most pernicious in finance because of the high levels of noise met in practice. We propose some tracks of research in order to improve the algorithms and results already obtained.

Keywords: *Generalisation performance, metric-based methods, model selection, regularization, approximate dynamic programming, reinforcement learning.*

[*] Département d'informatique et recherche opérationnelle, Université de Montréal, Montréal, Québec, Canada, H3C 3J7. Courriel : chapados@iro.umontreal.ca.

[†] CIRANO et Département d'informatique et recherche opérationnelle, Université de Montréal, Montréal, Québec, Canada, H3C 3J7, tél.: (514) 343-6804. Courriel: bengioy@iro.umontreal.ca.

Table des matières

1 Introduction

1.1 Aperçu du rapport

Ce rapport présente et propose plusieurs méthodes pour améliorer la *capacité de généralisation** des algorithmes d'apprentissage dans un contexte de prise de décisions financières. La présente introduction passe d'abord en revue les éléments fondamentaux de la théorie de l'apprentissage statistique (Vapnik 1998; Hastie, Tibshirani et Friedman 2001), incluant le concept crucial de la généralisation, qui est notre centre d'intérêt. Nous poursuivons avec un bref survol des champs d'application de l'apprentissage statistique en finance, et détaillons ensuite la pertinence des applications financières pour faire progresser le domaine de l'apprentissage statistique.

*Nous définissons tous les termes en italiques en temps opportun.

Le chapitre 2 résume un ensemble de résultats obtenus pendant la dernière année par le laboratoire lisa† sur les méthodes de *régularisation* et de *sélection de modèles* ; ces méthodes, globalement, visent à *contrôler la capacité* des algorithmes d'apprentissage en vue de limiter le problème du *sur-apprentissage*, qui est l'un des plus pernicieux en finance, à cause des niveaux d'incertitude élevés rencontrés en pratique. Nous proposons quelques pistes de recherche afin d'améliorer les algorithmes et résultats déjà obtenus.

†*Laboratoire d'informatique des systèmes adaptatifs, dirigé par Pr. Yoshua Bengio à l'Université de Montréal.*

Finalement, le chapitre 3 introduit une voie de recherche que nous entendons poursuivre au cours des prochains mois afin de résoudre le problème de la gestion optimale d'un *portefeuille d'options*. Ce problème‡ n'est étudié que depuis peu dans la littérature, et les résultats publiés jusqu'à maintenant sont, pour la plupart, fondés sur des suppositions très idéalistes (voire irréalistes) du fonctionnement des marchés financiers. Après avoir examiné les implications pratiques de ces suppositions, nous proposons une approche libre de presque toute hypothèse, et qui se formule naturellement comme un problème de programmation dynamique. Toutefois, pour tout problème de taille réaliste, l'infâme *malédiction de la dimensionalité* laissera voir sa griffe et sa dent féroces, et nous serons forcés de contrer ses assauts en faisant appel à des approximations basées sur les algorithmes d'apprentissage. Nous expliquons le détail des méthodes proposées, qui diffèrent en substance de celles présentement à l'état de l'art en programmation dynamique approximative. Ce troisième chapitre complète les résultats obtenus par l'équipe du lisa sur un problème apparenté de transaction d'options, résultats présentés dans un autre rapport cirano.

‡*Spécifiquement pour transiger des* ***options*** *en plus d'actions ordinaires.*

1.2 Définitions utiles en apprentissage statistique

1.2.1 Le cadre de l'apprentissage supervisé

Nous nous limitons dans le présent rapport au cadre de l'apprentissage supervisé, plus spécifiquement celui de la régression (Bishop 1995; Ripley 1996; Hastie, Tibshirani et Friedman 2001)*. Soit $X \in \mathbb{R}^p$ et $Y \in \mathbb{R}$ deux variables aléatoires dont la distribution jointe $P(X, Y)$ est fixe mais inconnue. Nous avons cependant accès à un ensemble de N paires $D = \{\langle x_i, y_i \rangle\}_{i=1}^N$, qu'on appelle *exemples d'entraînement*, qui sont tirées i.i.d.† de cette distribution.

*Nous dépassons quelque peu ce cadre au chapitre 3.

†Indépendants et identiquement distribués

Le problème de l'apprentissage supervisé pour la régression peut s'énoncer comme suit. Nous cherchons une fonction $f(X)$ pour prédire Y étant donné X. Plus formellement, soit $L(y, \hat{y})$ une *fonction de perte* qui pénalise les erreurs de prévision, où y est la valeur observée (« bonne réponse ») et $\hat{y}$ est la valeur prédite (qu'on souhaite la plus rapprochée possible de y). On cherche à identifier la fonction donnant l'*erreur de généralisation* minimale,

$$C(f) = E_{X,Y}[L(Y, f(X))]. \tag{1.1}$$

En théorie, la meilleure fonction qui résoud le problème de l'apprentissage supervisé est celle qui minimise cette erreur de généralisation,

$$f^* = \arg\min_{f \in \mathcal{F}} C(f), \tag{1.2}$$

où $\mathcal{F}$ représente la *classe de fonctions* à l'intérieur de laquelle on restreint la recherche. Nous considérons plus bas (§ 1.2.2) deux classes très importantes, celle des *fonctions linéaires* et celle des *réseaux de neurones artificiels*.

Il est d'une nécessité impérative d'accentuer le fait que les éq. (1.1) et (1.2) ne fournissent que des définitions théoriques, et **qu'on ne peut calculer exactement l'erreur de généralisation** dans le cadre que nous établissons : l'espérance de l'éq. (1.1) est prise par rapport à une distribution jointe $P(X, Y)$ que nous ne connaissons pas, même si nous effectuons toujours la supposition que cette distribution est fixée.

Fonction de perte pour la régression

La fonction de perte $L(y, \hat{y})$ habituellement retenue pour la *régression* (auquel cas Y peut assumer des valeurs continues dans un sous-ensemble de $\mathbb{R}$, plutôt qu'un nombre fini de valeurs discrètes, ce qui serait le cas pour un problème de *classification*) est l'*erreur quadratique*,

$$L(y, \hat{y}) = (y - \hat{y})^2. \tag{1.3}$$

On peut facilement montrer* que la fonction qui minimise l'erreur quadratique est aussi celle qui calcule l'*espérance conditionnelle* de Y étant donné X, ce qui est la « meilleure fonction » pour la majorité des problèmes de régression†,

$$\begin{aligned} C(f) &= E_{X,Y}[(Y - f(X))^2] \\ &= E_X\left[E_{Y|X}[(Y - f(X))^2|X]\right], \end{aligned}$$

et il suffit de prendre le $f(x)$ qui minimise en chaque point x,

$$\begin{aligned} f(x) &= \arg\min_c E_{Y|X}[(Y - c)^2|X = x] && (1.4) \\ &= \arg\min_c E_{Y|X}[Y^2 - 2cY + c^2|X = x] && (1.5) \\ &= E[Y|X = x]. && (1.6) \end{aligned}$$

*Par ex. (Hastie, Tibshirani et Friedman 2001).

†Il existe des cas ou, par exemple, une médiane conditionnelle est plus appropriée ; tout dépend de l'application et de la fonction de perte L.

Ensembles d'entraînement et de test

Tel que mentionné plus haut, il est pratiquement impossible de trouver une solution exacte au problème (1.2). On peut alors se contenter de trouver la meilleure fonction $\hat{f}$ sur l'ensemble d'entraînement D, et ainsi minimiser l'erreur d'entraînement, ou *perte empirique*,

$$C_E(f) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i)), \tag{1.7}$$

et dès lors

$$\hat{f} = \arg\min_{f\in\mathcal{F}} C_E(f). \tag{1.8}$$

Malheureusement, et c'est l'un des points les plus fondamentaux de la théorie de l'apprentissage statistique, la perte empirique n'est pas un bon estimateur de l'erreur de généralisation (Vapnik 1998) : c'est un estimateur **biaisé optimiste**, en ce sens qu'il sous-estime généralement l'erreur de généralisation. L'explication intuitive est que f est obtenu en résultat d'une minimisation, et par un résultat élémentaire en statistiques, nous avons pour tout ensemble de variables aléatoires i.i.d. $\{Z_1, Z_2, \ldots\}$,

$$E[\min(Z_1, Z_2, \ldots)] \leq E[Z_i],$$

d'où le biais.

Pour obtenir un estimateur non-biaisé de $C(f)$, on peut calculer la perte sur un *ensemble de test* disjoint de D, lui aussi tiré i.i.d. de la distribution sous-jacente $P(X, Y)$. L'erreur de test $\hat{C}(f)$ ainsi obtenue est un estimateur sans biais de $C(f)$ (mais qui n'est pas sans variance, laquelle dépend de la taille de l'ensemble de test).

1.2.2 Classes de fonctions

Un problème d'optimisation comme celui de l'éq. (1.8) doit normalement restreindre sa recherche à une classe de fonctions bien définie. Nous considérons ici des classes contenant des fonctions paramétrées par un vecteur de réels $\theta \in \mathbb{R}^P$: étant donné une classe et un vecteur de paramètres fixé, nous pouvons former une fonction.

Fonctions linéaires

Les fonctions linéaires sont parmi les plus simples. Soit le vecteur de paramètres $\theta = (\beta_0, \beta_1, \ldots, \beta_p)'$, la fonction calculée par le modèle linéaire est

$$f(x;\theta) = \beta_0 + \sum_{i=1}^{p} \beta_i \, x_i. \tag{1.9}$$

La dépendance en θ est souvent rendue explicite, tel que ci-haut.

Étant donné un ensemble d'entraînement $D = (\mathbf{X}, \mathbf{y})$ (où $\mathbf{X}$ est la matrice des entrées et $\mathbf{y}$ le vecteur des sorties désirées), on estime θ grâce, par exemple, à l'estimateur OLS* qui est bien connu en statistique,

$$\hat{\theta}^{\text{OLS}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}. \tag{1.10}$$

**Ordinary Least Squares; estimateur par moindres carrés ordinaire.*

Réseaux de neurones artificiels

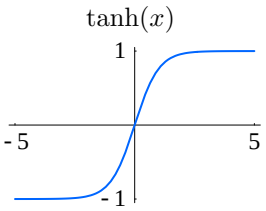
Une généralisation du modèle linéaire s'obtient en introduisant une étape intermédiaire non-linéaire entre les entrées et la sortie. Soit le vecteur de paramètres

$$\theta = (\alpha_{0,0}, \ldots, \alpha_{H,p}, \beta_0, \beta_1, \ldots, \beta_H)'.$$

La fonction calculée par un *réseau de neurones artificiels à H unités cachées*† est,

$$f(x;\theta) = \beta_0 + \sum_{i=1}^{H} \beta_i \tanh\left(\alpha_{i,0} + \sum_{j=1}^{p} \alpha_{i,j} x_i\right). \tag{1.11}$$

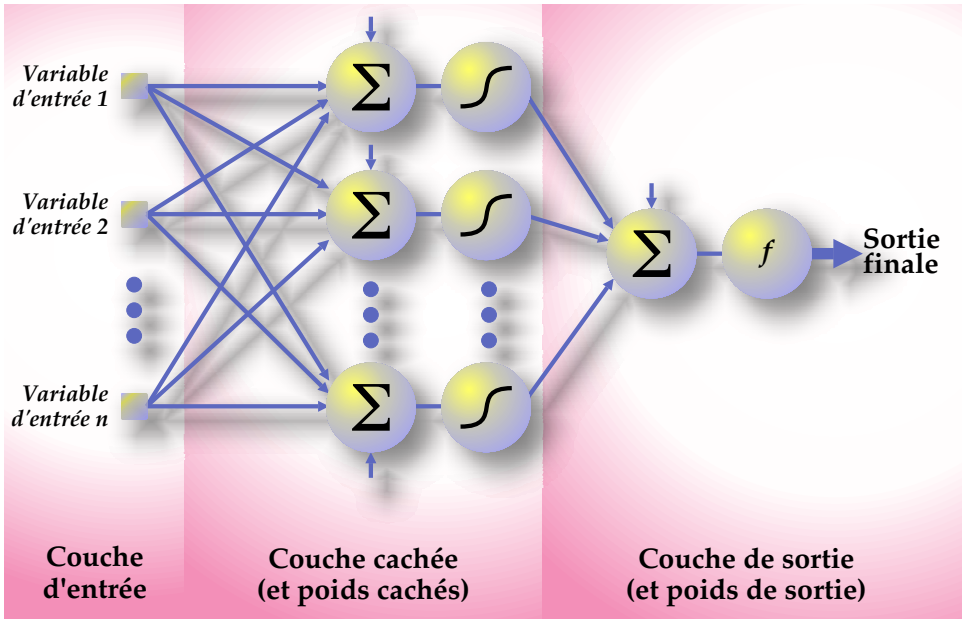
†*Qu'on appelle aussi* ***perceptron multi-couches****, ou de son acronyme anglais* MLP.



Un schéma général de l'allure d'un tel réseau apparaît à la figure 1.1. La fonction intermédiaire $\tanh(\cdot)$ introduit la non-linéarité dans le modèle, et est l'élément fondamental qui donne ses propriétes d'approximateur non-linéaire aux réseaux de neurones ; on peut montrer qu'un réseau avec suffisamment d'unités cachées H est un *approximateur universel*, en ce qu'il peut représenter n'importe quelle fonction continue sur un support compact avec une précision arbitraire‡.

‡*Voir* Hornik, Stinchcombe *et* White *(1989) pour la première démonstration de la propriété d'approximation universelle.*

Le vecteur de paramètres θ doit s'estimer par optimisation numérique en minimisant un critère de perte empirique tel que l'éq. (1.7). L'algorithme le plus employé pour calculer les gradients de cette fonction de perte par



◀ **Fig. 1.1.** *Schéma d'un réseau de neurones à propagation avant (perceptron multi-couches). Les non-linéarités de la couche cachée sont des fonctions Tanh. À chaque flèche est associée un poids multiplicatif faisant partie du vecteur de paramètres θ. La fonction de sortie f est arbitraire ; elle peut être l'identité pour la régression, ou une fonction logistique pour la classification.*

rapport à chaque paramètre du réseau est connu sous le nom de *rétropropagation* (Rumelhart, Hinton et Williams 1986; Bishop 1995), et procède par une application efficace des règles de dérivation en chaîne. Son utilisation s'effectue conjointement à des algorithmes d'optimisation non-linéaires classiques tels que la descente de gradient stochastique (Benveniste, Metivier et Priouret 1990) ou le gradient conjugué (Bertsekas 2000).

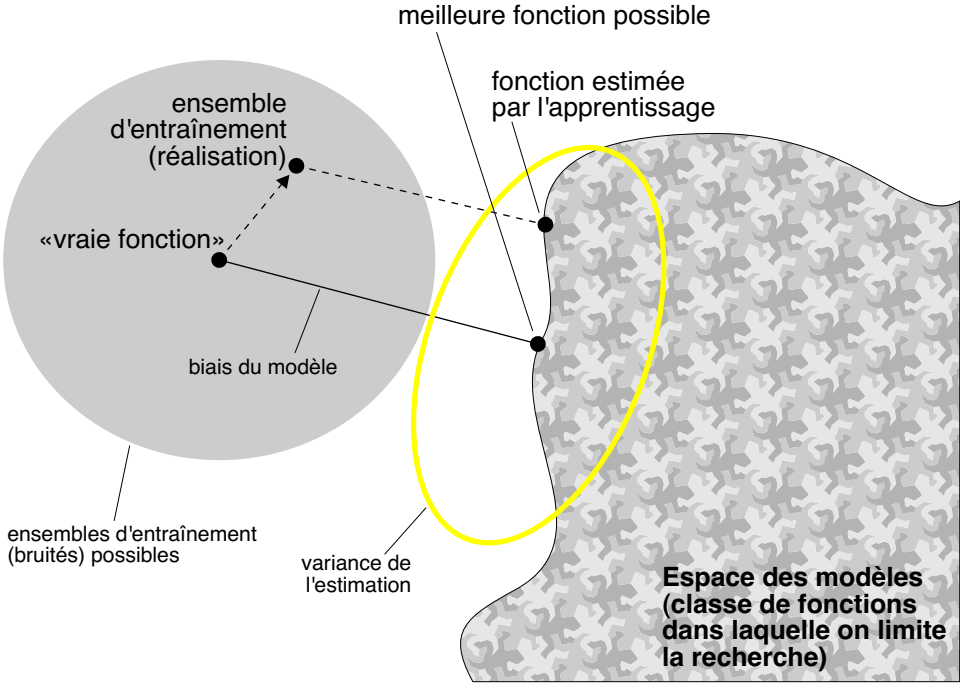
1.2.3 Le dilemme biais–variance

Pour un critère de perte quadratique, on peut décomposer l'erreur de généralisation en des termes explicites de *biais* et de *variance* qui fournissent beaucoup d'intuition sur les contributions respectives de différents facteurs à l'erreur globale. Ces décompositions sont classiques en statistique pour la régression linéaire, et ont été présentées dans le contexte des réseaux de neurones par Geman, Bienenstock et Doursat (1992).

Pour cette section, nous considérons que la sortie y est une fonction déterministe $f(\cdot)$ de l'entrée x, polluée par un bruit additif i.i.d. ϵ,

$$y = f(x) + \epsilon,$$

tel que $E[\epsilon] = 0$ et $E[\epsilon^2] = \sigma^2$. La fonction (aléatoire) trouvée par l'apprentissage $\hat{f}(\cdot)$ est celle qui minimise l'erreur quadratique sur un ensemble d'entraînement aléatoire. L'erreur quadratique de généralisation de $\hat{f}(\cdot)$ pour un x donné, en espérance sur tous les ensembles d'entraînement possibles,



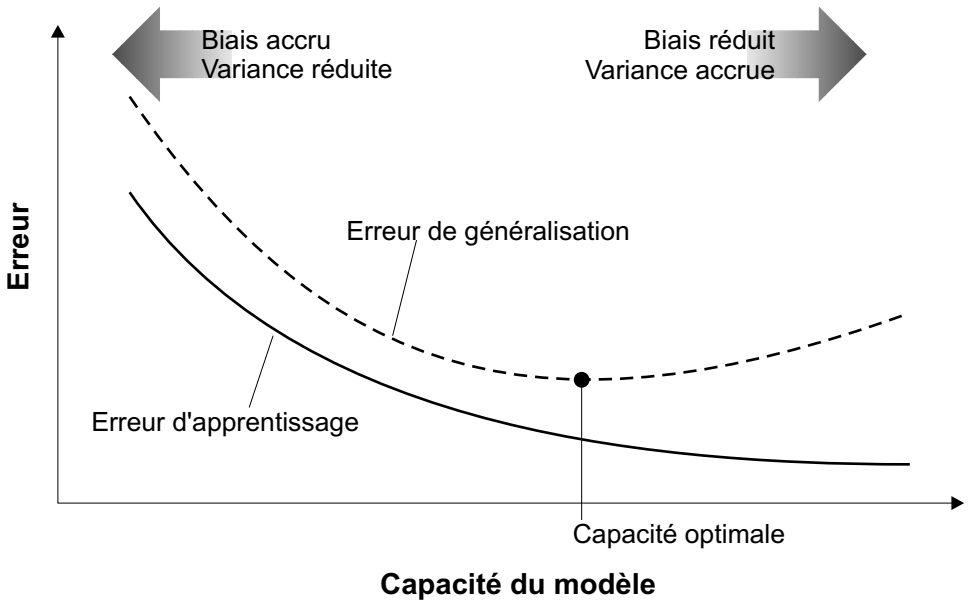
▶ **Fig. 1.2.** *Décomposition de l'erreur de généralisation en* ***biais*** *et* ***variance****. (Figure inspirée de* Hastie, Tibshirani *et* Friedman *(2001).)*

s'exprime par

$$\begin{aligned} E[(Y-\hat{f}(X))^2|X=x] &= E[(f(x)+\epsilon+E\hat{f}(x)-E\hat{f}(x)-\hat{f}(x))^2] \\ &= E[(\epsilon+(f(x)-E\hat{f}(x))+(\hat{f}(x)-E\hat{f}(x)))^2] \\ &= E[\epsilon^2]+E[(f(x)-E\hat{f}(x))^2] \\ &\quad +E[(\hat{f}(x)-E\hat{f}(x))^2] \\ &\quad \underbrace{-2E[(f(x)-E\hat{f}(x))(\hat{f}(x)-E\hat{f}(x))]}_{\text{zéro}} \\ &= \sigma^2+\underbrace{(f(x)-E\hat{f}(x))^2}_{\text{biais}^2}+\underbrace{E[(\hat{f}(x)-E\hat{f}(x))^2]}_{\text{variance}}. \end{aligned}$$

L'erreur à un point donné est donc fonction de trois composantes :

- Le bruit σ^2 à ce point, qui est une erreur irréductible.
- Le biais (carré) de la classe de fonctions, qui est la distance entre la « véritable » fonction $f(\cdot)$ et la meilleure fonction admissible dans la classe choisie $E\hat{f}(\cdot)$.
- La variance de la fonction estimée, qui représente la distance moyenne entre la fonction estimée $\hat{f}(\cdot)$ (qui dépend des particularités de l'ensemble aléatoire d'entraînement utilisé) et la « fonction moyenne » $E\hat{f}(\cdot)$.



◀ **Fig. 1.3.** *Une augmentation de la capacité d'un modèle fait toujours décroître l'erreur d'apprentissage, alors que l'erreur de généralisation augmente au-delà d'une* ***capacité optimale****, spécifique à chaque problème.*

Ces composantes de l'erreur de généralisation sont illustrés schématiquement à la figure 1.2. Le biais de l'erreur dépend strictement de la classe de fonctions choisie ; une classe plus riche (contenant plus de fonctions, et donc probablement une fonction plus rapprochée de la « vraie fonction ») aura un biais plus faible. En contrepartie, tout dépendant des aléas particuliers de l'ensemble d'entraînement utilisé*, la fonction estimée cherchera à se rapprocher de *cet ensemble d'entraînement*, et pourrait être très éloignée de la meilleure fonction à l'intérieur de la classe, c'est-à-dire celle correspondant à la projection de la vraie fonction sur la classe des fonctions admissibles. C'est le phénomène de la *variance*, qui est d'autant plus important que la classe de fonctions utilisée est riche†.

Ce compromis fondamental entre la richesse d'une classe de fonctions et la variance de l'estimation est baptisé « dilemme biais–variance » et est illustré schématiquement à la figure 1.3. Cette figure montre qu'à mesure que la richesse d'une classe de fonctions‡ augmente, l'erreur commise sur l'ensemble d'entraînement diminue de façon monotone. Simultanément, l'erreur de *généralisation* diminue initialement, conséquence d'une réduction du biais (car on se rapproche de la véritable fonction) ; cependant, au-delà d'un certain point, cette erreur *recommence à augmenter*, conséquence de l'augmentation de la variance (on apprend le bruit dans l'ensemble d'entraînement). Il existe donc un point de capacité optimale§ donnant la plus faible erreur de généralisation et qui correspond au meilleur compromis entre le biais et la variance ; ce point dépend de façon monotone croissante du nombre d'exemples

*Qui est de taille finie, et pollué par le bruit ϵ.

†*Car une classe riche aura plus d'occasions d'apprendre les réalisations du bruit dans l'ensemble d'entraînement.*

‡*Qu'on formalise mathématiquement grâce à la notion de* ***capacité de Vapnik–Chervonenkis*** *(*Vapnik *1998).*

§*Qui diffère pour chaque problème.*

N. La seule difficulté est que l'erreur d'entraînement, à elle seule, ne nous donne aucune indication pour trouver ce point ! Deux éléments émergent clairement :

1. Contrôle de la capacité Il est nécessaire de contrôler la capacité des modèles pour obtenir une bonne généralisation. Nous revenons amplement sur cet aspect au chapitre 2, qui traite de méthodes explicites de régularisation et de sélection de modèles pour ce faire.

2. Capacité optimale et validation croisée D'autre part, certaines méthodes (comme celles de régularisation que nous abordons au chapitre 2) offrent des outils de contrôle de la capacité, mais ne fournissent aucun moyen pour déterminer la capacité optimale, celle donnant la plus faible erreur de généralisation. Diverses techniques générales sont applicables dans ce cas, comme d'entraîner une variété de modèles de capacités différentes, et de choisir le meilleur selon l'erreur mesurée sur un *ensemble de validation**.

**C'est un autre ensemble disjoint des ensembles d'entraînement et de test, lui aussi tiré i.i.d. de $P(X, Y)$.*

Une généralisation particulièrement efficace de cette technique est la *validation croisée* (Stone 1974), qui divise l'ensemble d'entraînement D en un nombre K de sous-ensembles $\{D_i\}$,

$$D = \bigcup_{i=1}^{K} D_i, \quad D_i \cap D_j = \emptyset, i \neq j.$$

L'algorithme procède en répétant pour $i = 1, \ldots, K$,

1. Réserver l'ensemble D_i pour la validation.
2. Entraîner une variéte de modèles de capacités différentes sur l'ensemble d'entraînement $D - D_i$.
3. Tester ces modèles sur l'ensemble D_i et conserver le résultat.

Le modèle de « capacité optimale » (estimée) sélectionné est celui dont l'erreur de validation moyenne sur les ensembles $D_1, \ldots, D_K$ est la plus faible. Nous notons que les modèles sont toujours entraînés sur des données différentes de celles utilisées pour le test, et donc l'erreur de test ainsi calculée est un estimateur sans biais de l'erreur de généralisation†.

†*Cela ne dit rien de la variance de cet estimateur, qui peut être élevée, comme nous l'expliquons au chapitre 2.*

1.3 Comment l'apprentissage statistique profite de la finance

Nous justifions brièvement pourquoi un intérêt pour les applications financières est bénéfique pour faire progresser la théorie de l'apprentissage statistique.

Tout d'abord, la plupart des applications financières sont affligées d'un niveau de bruit élevé, dont la distribution exhibe des queues longues et épaisses, et parfois asymétriques (Campbell, Lo et MacKinlay 1997). Ce niveau de bruit élevé rend beaucoup de tâches de prévision, par exemple le rendement d'actifs, extrêmement difficiles.

Ensuite, bon nombre de séries sont mieux décrites par des modèles non-linéaires (Dunis et Zhou 1998) que les modèles linéaires utilisés traditionnellement en économétrie ; ces non-linéarités introduisent des dépendances sérielles complexes dans les séries chronologiques, qui violent l'hypothèse i.i.d. des innovations qui est généralement mise de l'avant. La modélisation adéquate de ces dépendances demeure un défi, et est rendue difficile par les niveaux de bruit élevés notés plus haut.

Par ailleurs, et cette question demeure en grande partie ouverte pour l'apprentissage statistique, les séries économétriques sont *non-stationnaires* (Hackl 1989; Campbell, Lo et MacKinlay 1997) ; le modèle qui les décrit change avec le temps, soit à cause d'ajustements progressifs de l'économie ou de chocs structuraux brutaux. En contraste, presque toute la théorie de l'apprentissage statistique est fondée sur l'hypothèse que la distribution sous-jacente, même si elle est inconnue, demeure fixe. Les méthodes actuelles pour tenir compte de ces non-stationarités sont pour la plupart *ad hoc**.

**Par exemple, l'utilisation d'un pas de gradient constant dans une optimisation à base de gradient stochastique.*

Finalement, et cet aspect est souvent ignoré, beaucoup de problèmes en finance n'ont pas de formulation simple en termes classiques pour l'apprentissage statistique. Par exemple, un agent économique ne cherche à faire ni de la classification ni de la régression ; c'est—au sens orthodoxe—une machine à *maximiser son utilité*. Il est donc nécessaire de reformuler spécifiquement pour la finance les critères d'optimisation de l'apprentissage, qui seraient autrement inadaptés.

2 Méthodes de régularisation et méthodes métriques

Ce chapitre traite de quelques méthodes offrant un contrôle explicite de la capacité, afin d'obtenir le meilleur compromis entre le biais et la variance pour chaque problème. Elles entrent dans la famille générale des méthodes de sélection et de combinaisons de modèles, un aperçu récent desquelles est présenté dans un numéro spécial de la revue *Machine Learning* (Bengio et Schuurmans 2002).

Les méthodes de **sélection de modèles** considèrent une grande classe de fonctions* $\mathcal{F}$, et la décomposent en un ensemble discret de sous-classes, par exemple $\{\mathcal{F}_0, \mathcal{F}_1, \ldots \subseteq \mathcal{F}\}$. Étant donné des données d'entraînement, ces méthodes tentent ensuite d'identifier la sous-classe optimale de laquelle choisir l'hypothèse finale, celle promettant la meilleure performance de généralisation. Une variété de méthodes existe pour ce faire ; certaines sont basées sur une *pénalisation de la complexité de la classe de fonctions* (tels que le critère de *minimum description length* (Rissanen 1986), ou encore certains classiques en statistiques qui pénalisent le nombre de paramètres ajustables (Akaike 1973; Foster et George 1994)). D'autres méthodes de sélection font appel aux données elles-mêmes, comme la validation croisée décrite précédemment (Stone 1974) ou le bootstrap (Efron 1979). Nous considérons plus bas (§ 2.3) une nouvelle famille de méthodes de sélection basées sur un critère métrique.

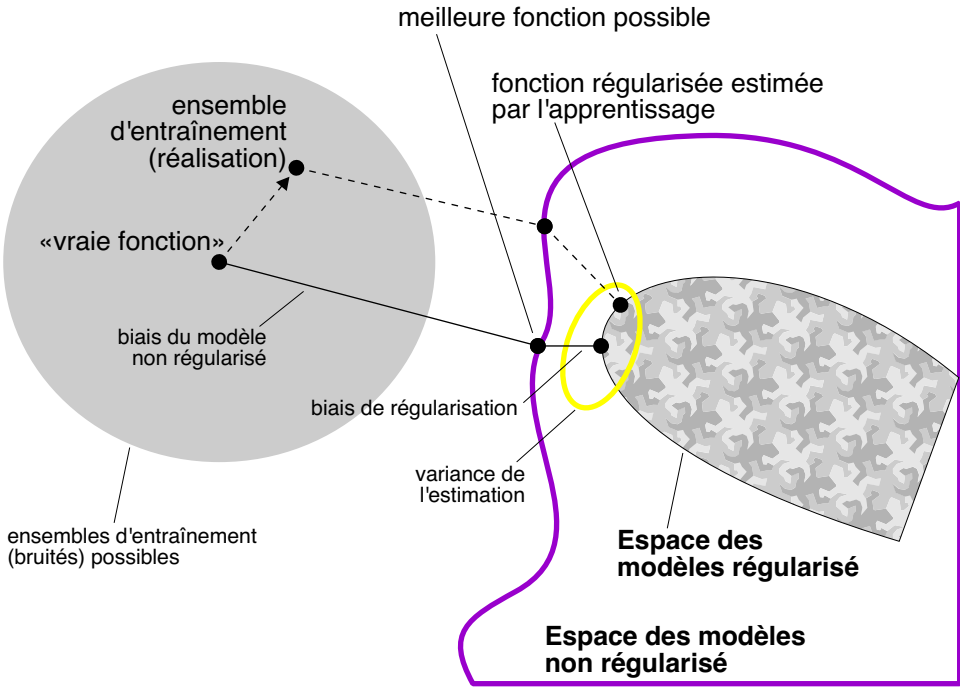
*Rappelons qu'une **classe de fonctions** est un ensemble de fonctions parmi lesquelles un algorithme d'apprentissage est contraint à trouver une solution.*

Nous présentons d'abord les méthodes de régularisation, qui ne supposent pas une décomposition discrète de la classe de fonctions, mais imposent plutôt une pénalisation aux fonctions individuelles.

2.1 Qu'est-ce que la régularisation ?

De manière générale, les méthodes de régularisation optimisent un critère d'apprentissage légèrement différent de celui servant à évaluer l'erreur de généralisation. Ce critère modifié introduit une préférence sur certaines solutions jugées meilleures à priori ; par exemple, entre deux solutions optimisant également bien un critère principal† la régularisation permettra de favoriser la solution la plus « simple » (restant à définir).

†*Par exemple, l'erreur quadratique pour un problème d'estimation de l'espérance conditionnelle.*



▶ **Fig. 2.1.** *La régularisation contraint la classe de fonctions admissibles, et permet une grande réduction de la variance de l'estimation au prix d'une (légère, on l'espère) augmentation du biais.*

Certaines de ces méthodes sont bien connues en statistiques, sous les noms de régression *ridge* (Hoerl et Kennard 1970) et *lasso* (Tibshirani 1996). Respectivement, le *ridge* correspond à pénaliser la norme L_2 des coefficients $\theta = (\theta_1, \ldots, \theta_P)'$ d'un modèle,

$$C_{\text{ridge}}(\theta, \lambda) = C_E(\theta) + \lambda \sum_{i=1}^{P} \theta_i^2, \tag{2.1}$$

alors que le *lasso* impose une contrainte sur la norme L_1 des coefficients,

$$C_{\text{lasso}}(\theta, \lambda) = C_E(\theta) \tag{2.2}$$

$$\text{sous la contrainte} \quad \sum_{i=1}^{P} |\theta_i| \leq \lambda. \tag{2.3}$$

Dans ces équations $C_E(\theta)$ est la fonction objective originale à optimiser sur l'ensemble d'entraînement*. Dans les deux cas, le paramètre λ contrôle l'importance relative du terme de régularisation dans la fonction globale à optimiser.†

La figure 2.1 illustre la conséquence de la régularisation en termes des concepts de biais et de variance explicités au chapitre 1. Au prix de ce qu'on espère être une faible augmentation du biais, les méthodes de régularisation

*Celle servant ultimement à mesurer l'erreur de généralisation, par exemple, l'erreur quadratique pour la régression

†Ce genre de paramètre se nomme **hyperparamètre** car il n'est pas optimisé directement en même temps que les composantes de θ.

permettent une réduction de la variance de l'estimation, de telle sorte que (pour les hyperparamètres correctement sélectionnés) l'erreur de généralisation totale s'en trouve réduite.

Il est par ailleurs possible de donner une *interprétation bayesienne* à ces termes de régularisation, qui correspond à un *à priori* sur la distribution des coefficients. On peut montrer que la pénalisation *ridge* est équivalente à supposer un à priori gaussien sur les paramètres, alors que la pénalisation *lasso* équivaut à un à priori laplacien.

Dans le domaine des algorithmes d'apprentissage, la pénalisation de type *ridge* porte le nom de *weight decay*—pénalisation sur la norme des poids—(Hinton 1986; Bishop 1995), mais s'exprime identiquement. En pratique, il est courant d'omettre certains paramètres dans la pénalisation, tels que ceux associés aux termes de biais dans des réseaux de neurones artificiels, qu'on désire estimer sans biais.

L'intérêt pragmatique pour ces méthodes s'est accru avec les résultats de Weigend, Rumelhart et Huberman (1991) qui ont montré empiriquement une amélioration de la capacité de généralisation. Cette amélioration s'interprète aisément en termes d'une réduction significative de la variance des modèles estimés, tel que décrit précédemment.

2.2 Pénalisation sur la norme des entrées

2.2.1 Description de la méthode

La pénalisation sur la norme des entrées est une méthode de sélection « douce » de variables par régularisation, étudiée pour la première fois par Chapados (2000) et Chapados et Bengio (2001a) dans le cadre d'un système intégré de décision financière. Au contraire des méthodes combinatoires comme les procédures de *branch and bound* (Narendra et Fukunaga 1977) et la sélection avant ou arrière (Bishop 1995), cette méthode ne cherche pas « le bon » sous-ensemble des entrées à fournir au réseau ; toutes les entrées sont fournies, et la méthode pénalisera automatiquement les connections dans le réseau provenant des entrées qui ne s'avèrent pas importantes*.

*Nous présentons ici la méthode dans le cadre des perceptrons multi-couches, mais elle s'applique également à des modèles similaires.

La pénalisation sur la norme des entrées est similaire à celle sur la norme des poids présentée ci-haut en ce qu'elle ajoute un terme à la fonction de coût. Elle pénalise les poids qui relient une entrée particulière du réseau à toutes les unités cachées, en agissant comme suit : soit $\theta^{(1)}_{jh}$ le poids (situé sur la première couche du MLP) unissant l'entrée j à l'unité cachée h, la

pénalisation attribuée à l'entrée j est,

$$C_{\mathrm{ID}}^{(j)}(\theta) = \sum_{h=1}^{H} \left(\theta_{jh}^{(1)}\right)^2, \tag{2.4}$$

où H est le nombre d'unités dans la première couche cachée. La figure 2.2 illustre les poids qui font partie de $C_{\mathrm{ID}}^{(j)}(\theta)$.

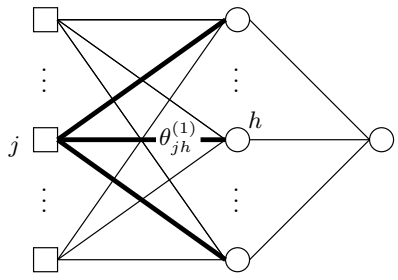
Pour déterminer la contribution complète $C_{\mathrm{ID}}(\theta)$ à la fonction de coût, nous faisons une somme des contributions non-linéaires de toutes les entrées j, comme suit,

$$C_{\mathrm{ID}}(\theta, \lambda, \eta) = \lambda \sum_{j} \frac{C_{\mathrm{ID}}^{(j)}}{\eta + C_{\mathrm{ID}}^{(j)}(\theta)}, \tag{2.5}$$

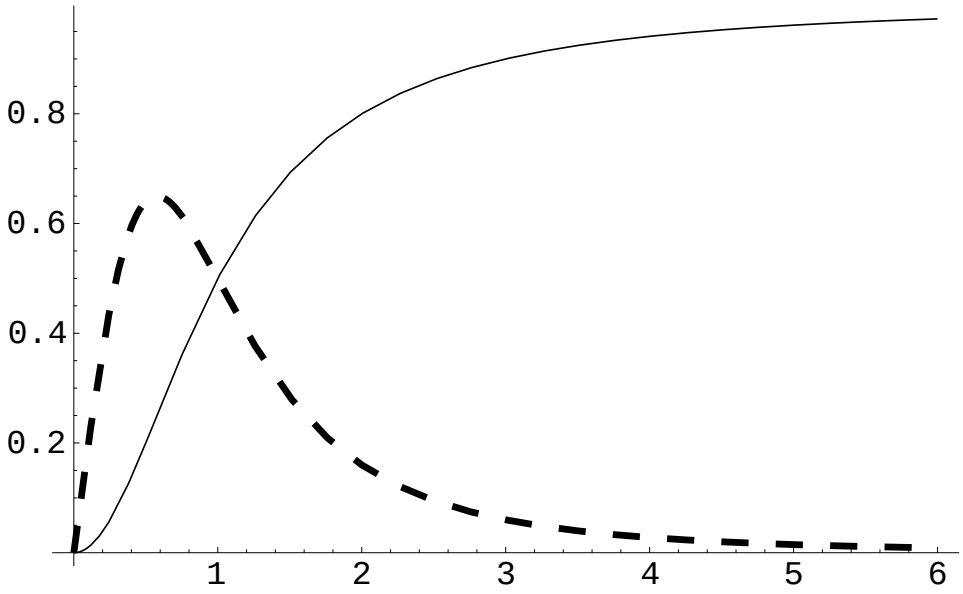
et λ, qui est un hyperparamètre, contrôle l'importance relative de la pénalisation sur la norme des poids. De même, η est un autre hyperparamètre qui agit comme *seuil de saturation* ; la figure 2.3 illustre l'allure de la fonction $x^2/(\eta + x^2)$, qui module la contribution de chaque entrée. On constate que pour de petites valeurs de la norme des poids, la fonction se comporte de manière quadratique, mais elle sature rapidement dès que cette norme dépasse le seuil déterminé par η.

Nous pouvons comprendre l'effet de cette fonction comme suit : initialement lors de l'entraînement, lorsque les poids émergeant d'une entrée j sont encore petits, le réseau doit subir un coût marginal élevé pour en accroître la grandeur ; si l'entrée ne s'avère utile que pour peu d'unités cachées ou peu d'exemples, il est préférable de laisser ces poids petits, et donc d'éliminer l'impact pratique de cette entrée sur le réseau. En contrepartie, dès le moment où l'entrée prouve son utilité (avec des poids suffisamment grands), la contribution de ces poids à la fonction de coût devient constante, indépendamment de leurs valeurs.

Cette méthode est similaire à celle de *weight elimination* proposée par Weigend, Rumelhart et Huberman (1991) en ce qu'elle utilise une fonction de coût de forme similaire à l'éq. (2.5). Cependant, le *weight elimination*



◀ **Fig. 2.2.** *Schéma des connections affectées par la pénalisation sur la norme des entrées pour une entrée j dans un perceptron multi-couches (gras) ; la pénalisation est la plus faible lorsque* ***tous les poids*** *associés à l'entrée j sont réduits à zéro.*



◀ **Fig. 2.3.** *Fonction de coût $x^2/(\eta + x^2)$ appliquée par la pénalisation sur la norme des entrées à la norme des poids reliant une entrée donnée du réseau à toutes les unités cachées (trait continu) et dérivée première (pointillé), pour $\eta = 1$.*

n'est pas habituellement employé pour faire de la sélection de variables ; il s'applique à tous les poids du réseau.

2.2.2 Contributions au domaine

Chapados et Bengio (2001a) ont observé un apport statistiquement très significatif du terme de régularisation sur la norme des entrées à la performance financière mesurée hors-échantillon d'un système de gestion de portefeuille. Plus récemment, Chapados et Bengio (2001b) ont validé ces résultats à des problèmes de régression linéraire difficiles générés artificiellement. Le cadre expérimental exact est décrit dans l'article précité ; brièvement, nous considérons des problèmes de régression avec 30 variables d'entrées (lesquelles sont potentiellement hautement corrélées) et seulement 60 exemples d'entraînement*, avec un niveau de bruit élevé sur la variable dépendante. Nous comparons l'emploi de la pénalisation sur la norme des entrées (conjointement avec l'algorithme ADJ décrit en § 2.3.2) aux méthodes suivantes :

1. Régression ordinaire par moindres carrés, sans régularisation.
2. Sélection de variables avant, qui consiste à introduire une-à-une les variables d'entrées, dans l'ordre (myope) qui réduit le plus l'erreur.

Les résultats obtenus sont très prometteurs : ils démontrent que la performance de généralisation (hors-échantillon) de la régression effectuée avec la pénalisation sur la norme des entrées est **statistiquement significativement bien meilleure** que les deux autres méthodes considérées†.

** Ce qui serait considéré quasi-hérétique pour une régression classique.*

† Les p-valeurs sont presque toutes inférieures à 0.001, ce qui est indicatif d'un niveau très élevé de significativité statistique.

2.2.3 Problèmes ouverts

Certaines questions demeurent toutefois en suspens. Tout d'abord, le problème le plus délicat demeure celui des minima locaux qui sont introduits dans la fonction de coût ; nous constatons empiriquement que les réseaux entraînés avec la pénalisation sur la norme des entrées sont plus susceptibles de sombrer dans de mauvais minima locaux au cours de l'entraînement, ce qui rend l'application pratique de la méthode moins attrayante. Nous devons évaluer si l'emploi d'algorithmes d'optimisation plus robustes à ces minima* permettra d'amoindrir ou d'aplanir ces difficultés.

*Tels que ceux à base de gradient stochastique, dans le cadre de l'entraînement de réseaux de neurones.

Ensuite, nous devons caractériser précisément l'effet de cette pénalisation d'un point de vue théorique et empirique. Il apparaît clairement, d'après nos expériences, que l'emploi de cette pénalisation ne mène pas à une réduction totale des poids associées aux entrées inutiles : observe-t-on ici une conséquence des obstacles d'optimisation évoqués ci-haut, ou une manifestation plus profonde de l'effet de la méthode† ? De plus, est-ce que cette pénalisation peut s'exprimer assez simplement comme un à priori bayesien sur les paramètres du réseau ?

†*Nous savons déjà, par exemple, que la régression* ridge *a le même effet, alors que le* lasso *conduit les paramètres inutiles vers zéro.*

Finalement, il reste à comprendre comment cette méthode peut se combiner aux méthodes d'élaguage des poids, comme le *Optimal Brain Damage* (Le Cun, Denker et Solla 1990) ou le *Optimal Brain Surgeon* (Hassibi et Stork 1993), lesquelles entreprennent d'éliminer les poids inutiles sur la base d'une théorie distributionnelle‡ des poids résultant de l'entraînement.

‡*Souvent en simple approximation quadratique.*

2.3 Méthodes métriques

Dans beaucoup d'applications des algorithmes d'apprentissage, il est très facile et peu coûteux d'acquérir de grandes quantités de données pour lesquelles aucune « bonne réponse » n'est donnée. Par exemple, il existe abondance d'images satellites sans catégorisation, ou d'heures et d'heures d'enregistrement de conversations sans transcription. Ces données ne sont pas directement utiles à des algorithmes d'apprentissage supervisé, qui ont besoin de *valeurs cibles* pour leur entraînement. Cependant, elles peuvent être très instructives sur la distribution des situations qui seront rencontrées en pratique, et leur abondance peut mettre en lumière des cas qui sont absents des données d'entraînement (supervisées) proprement dites. Les *méthodes métriques* que nous considérons maintenant cherchent à exploiter la richesse de tels ensembles non-étiquettés afin d'améliorer la généralisation.

Les méthodes métriques pour la sélection de modèles et la régularisation sont fondées sur une intuition simple : les solutions qui résultent d'un *sur-apprentissage* (« overfitting ») sont susceptibles de se comporter très

différemment sur les points d'entraînement par rapport aux points « voisins ». Cela se produit car le critère d'optimisation cherche à réduire l'erreur précisément sur les points d'entraînement, mais n'impose autrement aucune contrainte à la solution. Cependant, pour obtenir une bonne généralisation, nous cherchons à représenter adéquatement la fonction sous-jacente non seulement aux points d'entraînement, mais en général à tous les points pour lesquels la distribution des entrées $P(X)$ est significative. La figure 2.4 illustre la situation.

Ces méthodes sont toutes basées sur la définition d'une (pseudo) *métrique* $d(f, g)$ sur l'espace des fonctions, permettant de juger de la distance entre deux fonctions. Nous exigerons la satisfaction des trois axiomes suivants :

1. **Nonnégativité**, $d(f, g) \geq 0$.
2. **Symétrie**, $d(f, g) = d(g, f)$.
3. **Inégalité triangulaire**, $d(f, g) \leq d(f, h) + d(h, g)$.

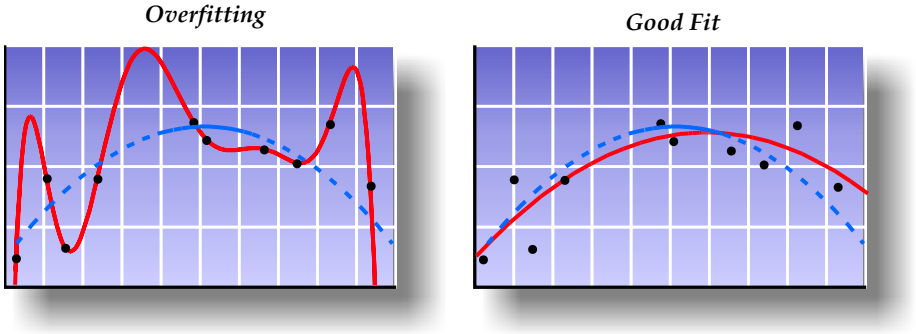
Nous considérons le cadre usuel de l'apprentissage supervisé, dans lequel les exemples d'entraînement $\{\langle x_i, y_i \rangle\}$ sont tirés i.i.d. d'une distribution stationnaire fixe mais inconnue $P(X, Y)$ sur $X \times Y$. Dans ce cadre, une mesure appropriée de la distance entre deux fonctions est donnée par

$$d(f, g) = \psi\left(E\left[L(f(X), g(X))\right]\right), \tag{2.6}$$

où l'espérance $E[\cdot]$ est prise par rapport à la distribution des entrées $P(X)$, et $\psi(\cdot)$ est une fonction de normalisation nécessaire à l'obtention d'une métrique. Un exemple approprié à la régression utilise une fonction de perte quadratique $L(u, v) = (u - v)^2$, et la normalisation donnée par $\psi(z) = z^{1/2}$.

Schuurmans (1997) a proposé d'estimer $d(f, g)$ empiriquement sur des *données non-étiquettées*, c'est-à-dire des points $\{\tilde{x}_i\}_{i=1}^{U}$ tirés i.i.d. de $P(X)$ mais pour lesquels aucun y_i n'est donné ; il s'agit, bien sûr, d'un estimateur de Monte Carlo classique de l'éq. (2.6). Dans ce qui suit, nous dénotons $\tilde{d}(f, g)$ l'*estimateur* de $d(f, g)$, tel que calculé sur un grand ensemble de données non-étiquettées*,

*Nous supposons que la variance de cet estimateur est suffisamment petite.



◀ **Fig. 2.4.** *Illustration du* ***sur-apprentissage****.* ***Gauche*** *: La solution (trait plein) apprend le bruit des données (points noirs), mais pas la structure sous-jacente (pointillée).* ***Droite*** *: La solution moins flexible évite le problème.*

$$\tilde{d}(f,g) = \psi\left(\frac{1}{U}\sum_{i=1}^{U} L(f(\tilde{x}_i), g(\tilde{x}_i))\right). \tag{2.7}$$

Par opposition, la distance entre deux fonctions, estimée seulement sur *l'ensemble d'entraînement* $D = \{\langle x_i, y_i\rangle\}_{i=1}^{\ell}$, est dénotée (remarquez qu'elle n'utilise pas les cibles (variables dépendantes) y_i de l'ensemble d'entraînement)

$$\hat{d}(f,g) = \psi\left(\frac{1}{\ell}\sum_{i=1}^{\ell} L(f(x_i), g(x_i))\right). \tag{2.8}$$

Nous introduisons de plus la notation suivante pour comparer une fonction f à la « véritable fonction » sous-jacente, que nous ne connaissons pas,

$$d(f, P(Y|X)) = \psi\left(E\left[L(f(X), Y)\right]\right), \tag{2.9}$$

où l'espérance est prise sur la distribution jointe $P(X,Y)$. La distance estimée seulement à partir des points de l'ensemble d'entraînement est notée par

$$\hat{d}(f, P(Y|X)) = \psi\left(\frac{1}{\ell}\sum_{i=1}^{\ell} L(f(x_i), y_i)\right). \tag{2.10}$$

Ceci n'est évidemment rien d'autre que l'erreur d'entraînement ; nous utilisons cette notation particulière seulement dans la présente section sur les méthodes métriques.

2.3.1 L'algorithme TRI

L'algorithme TRI (Schuurmans 1997; Schuurmans et Southey 2001) est un algorithme de *sélection de modèles* qui est basé sur l'idée d'exploiter la géométrie naturelle de la distribution des vecteurs d'entrée pour identifier quels modèles (parmi un ensemble donné à l'algorithme) sont les plus « crédibles ». Pour ce faire, il cherche à identifier des violations de l'*identité triangulaire** d'après les mesures de distances définies précédemment. L'algorithme suppose la considération d'une suite partiellement ordonnée de classes de fonctions de complexité croissante $\{\mathcal{F}_0, \mathcal{F}_1, \ldots, \mathcal{F}_K\}$. L'algorithme n'exige que la définition d'un ordre partiel sur cette suite, mais pour simplifier la présentation, nous supposerons l'imposition d'un ordre total,

$$\mathcal{F}_0 \subseteq \mathcal{F}_1 \subseteq \cdots \subseteq \mathcal{F}_K.$$

Par exemple, la suite de classes constituée des réseaux de neurones avec un nombre *fixé* d'unités cachées, mais entraînés avec des coefficients *décroissants* de pénalisation sur la norme des poids (*weight decay*) satisfait à un tel ordre†.

* *D'où l'algorithme tire son nom.*

† *Un gros coefficient de pénalisation encourage les petits poids dans le réseau, ce qui réduit la propension à atteindre certaines fonctions au cours de l'entraînement.*

L'algorithme TRI se voit fourni un ensemble de fonctions (qu'on nomme « hypothèses ») $\{f_0, f_1, \ldots, f_K\}$ résultant de l'entraînement sur l'ensemble D, chacune optimisant le critère d'entraînement (par exemple, l'erreur quadratique) sur sa classe de fonctions respective, $\{\mathcal{F}_0, \mathcal{F}_1, \ldots, \mathcal{F}_K\}$. Il doit, rappelons-le, choisir *la bonne fonction* qui semble présenter les meilleures garanties de généralisation, c'est-à-dire celle avec une faible erreur d'entraînement, **mais qui n'a pas sur-appris** les données. Évidemment, l'erreur d'entraînement à elle seule n'est pas suffisante pour détecter le surapprentissage, car une fonction issue d'une classe riche aura toujours une plus petite erreur d'entraînement qu'une autre issue d'une classe plus restreinte qui est un sous-ensemble de la première. En d'autres termes, nous nous attendons, *à priori*, à observer la relation suivante étant donné l'ordre total introduit ci-haut,

$$\hat{d}(f_j, P(Y|X)) \leq \hat{d}(f_k, P(Y|X)), \quad \forall j \geq k.$$

Considérons deux hypothèses successives f_k et f_{k+1}, qui ont toutes deux une faible erreur d'entraînement, mais ont une grande distance entre elles. Par simple géométrie, il est impossible que deux fonctions très distantes l'une de l'autre soient simultanément proches de la « véritable fonction ». Dans cette circonstance, nous serions portés à retenir la première hypothèse et rejeter la seconde, car cette dernière, étant plus complexe, est plus susceptible au surapprentissage. En fait, si $\hat{d}(f_k, P(Y|X))$ et $\hat{d}(f_{k+1}, P(Y|X))$ sont tous deux de bons estimateurs de l'erreur de généralisation, ils doivent *nécessairement obéir à l'inégalité triangulaire* avec la distance « connue » $\tilde{d}(f_k, f_{k+1})^*$, *viz.*,

$$\hat{d}(f_k, P(Y|X)) + \hat{d}(f_{k+1}, P(Y|X)) \geq \tilde{d}(f_k, f_{k+1}).$$

*Estimée avec un **très grand** ensemble de données non-étiquettées, cf. éq. (2.7).*

L'algorithme TRI exploite formellement cette intuition : il choisit dans la séquence $\{f_0, \ldots, f_K\}$ la fonction f_k d'indice k le plus élevé pour laquelle l'identité triangulaire

$$\hat{d}(f_k, P(Y|X)) + \hat{d}(f_j, P(Y|X)) \geq \tilde{d}(f_k, f_j), \quad \forall j \leq k \tag{2.11}$$

est satisfaite avec toutes les fonctions précédentes f_j dans la séquence.

2.3.2 L'algorithme ADJ

L'algorithme ADJ (SCHUURMANS et SOUTHEY 2001) en est aussi un de sélection de modèles, fondé sur des intuitions similaires à TRI. ADJ exploite les données non-étiquettées pour *ajuster*† les erreurs d'entraînement de chaque modèle afin de tenter de les rapprocher de l'erreur de généralisation. La sélection de modèles est ensuite effectuée simplement en fonction du modèle ayant l'erreur ajustée la plus faible.

†*Ajustement qui lui donne son nom.*

D'un point de vue opérationnel, l'erreur d'entraînement ajustée pour une hypothèse f_k (notée $\hat{\hat{d}}(f_k, P(Y|X))$) se calcule ainsi par ADJ,

$$\hat{\hat{d}}(f_k, P(Y|X)) = \hat{d}(f_k, P(Y|X)) \max_{j<k} \frac{\tilde{d}(f_j, f_k)}{\hat{d}(f_j, f_k)}. \tag{2.12}$$

En d'autres termes, l'erreur d'apprentissage de f_k est ajustée par un facteur qui est fonction du ratio maximum observé entre la « véritable » distance entre f_k et ses prédécesseurs, et la distance mesurée sur l'ensemble d'entraînement seulement. Cet ajustement semble quelque peu *ad hoc*, mais SCHUURMANS et SOUTHEY (2001) démontrent le résultat suivant, qui établit que ADJ ne sous-estime pas l'erreur de généralisation par un facteur beaucoup plus grand que 3, et borne donc la magnitude du sur-apprentissage auquel l'algorithme est sujet.

Proposition 2.1 (Schuurmans) *Soit f_m l'hypothèse optimale dans la séquence $f_0, f_1, \ldots$, et soit f_k l'hypothèse retenue par* ADJ. *Si (i) $m \leq k$, et (ii) $\hat{\hat{d}}(f_m, P(Y|X)) \leq d(f_m, P(Y|X))$ alors*

$$d(f_k, P(Y|X)) \leq \left(2 + \frac{\hat{d}(f_m, P(Y|X))}{\hat{d}(f_k, P(Y|X))}\right) d(f_m, P(Y|X)). \tag{2.13}$$

Une autre proposition fournit aussi une assurance (faible) contre le *sous-apprentissage* (« underfitting »).

Proposition 2.2 (Schuurmans) *Considérons une hypothèse f_m, et supposons que (i) $\hat{d}(f_k, P(Y|X)) \leq d(f_k, P(Y|X))$ pour tout $0 \leq k \leq m$, et (ii) $\hat{d}(f_k, P(Y|X)) \leq \hat{\hat{d}}(f_k, P(Y|X))$ pour tout $0 \leq k \leq m$. Alors, si nous avons de plus*

$$d(f_m, P(Y|X)) < \frac{1}{3} \frac{\hat{d}(f_k, P(Y|X))^2}{d(f_k, P(Y|X))}, \tag{2.14}$$

pour tout $0 \leq k < m^*$, il s'ensuit que

$$\hat{\hat{d}}(f_m, P(Y|X)) < \hat{\hat{d}}(f_k, P(Y|X)) \tag{2.15}$$

pour tout $0 \leq k < m$, et donc ADJ *choisira f_m plutôt qu'un prédécesseur.*

*En d'autres termes, $d(f_m, P(Y|X))$ est suffisamment petit.

Intérêt de ces propositions Elles démontrent que ADJ peut s'analyser en termes de bornes pire-cas contre le sur- et le sous-apprentissage, étant données certaines suppositions†. De telles bornes ne peuvent pas être établies pour les méthodes conventionnelles de sélection de modèles dans le cas général, i.e. sans faire d'hypothèses distributionnelles sur les données.

†SCHUURMANS *et* SOUTHEY *(2001) font cependant remarquer que ces suppositions ne sont toutefois pas toujours observées en pratique.*

2.3.3 Contributions au domaine : lorsqu'on n'a pas de données non-étiquettées

Récemment, Bengio et Chapados (2001) ont proposé une extension simple aux algorithmes TRI et ADJ*, applicable au cas où **aucun ensemble non-étiquetté n'est donné**. Sous-tendant cette extension est l'idée qu'une fonction qui généralisera adéquatement devra se comporter sur les points d'entraînement d'une manière qui est similaire aux points *qui leurs sont voisins*. Nous avons donc proposé de remplacer l'estimateur $\tilde{d}(f, g)$ par un autre qui est calculé depuis un *estimateur* $\hat{P}(X)$ *de la distribution des entrées*,

$$\bar{d}(f, g) = \psi\left(E_{\hat{P}(X)}\left[L(f(X), g(X))\right]\right). \tag{2.16}$$

L'estimateur $\hat{P}(X)$ est construit strictement à partir des données disponibles dans l'ensemble d'entraînement, sans faire appel à un ensemble non-étiquetté séparé. Dans nos expériences, nous avons fait appel à un estimateur non paramétrique simple, les fenêtres de Parzen (décrites plus bas), mais des méthodes plus sophistiquées† sont facilement envisageables.

Il est à noter que l'estimateur de distance $\bar{d}(f, g)$ se calcule parfois analytiquement, si la forme fonctionnelle de l'estimateur $\hat{P}(X)$ s'y prête (ce qui est le cas pour les fenêtres de Parzen). De façon alternative, nous pouvons échantillonner un nouvel ensemble non-étiquetté de la distribution $\hat{P}(X)$ et utiliser un estimateur de distance calculé comme $\tilde{d}(f, g)$.

*De même que ADA que nous considérons plus bas.

†*Qui incorporent des connaissances à priori sur le problème, par exemple.*

Fenêtres de Parzen

Les fenêtres de Parzen (Parzen 1962) sont une forme simple d'estimation *non-paramétrique*‡ de densités. La méthode consiste à placer une fonction de « noyau » $\varphi(\cdot)$ autour de chaque exemple d'entraînement $\{x_i\}_{i=1}^N$, et à considérer la somme pondérée (et normalisée) de ces noyaux comme l'approximateur de densité (en supposant qu'il existe une densité $p_X(x)$ correspondant à la distribution $P(X)$),

$$\hat{p}_X(x) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{V_N}\varphi\left(\frac{x - x_i}{h_N}\right), \tag{2.17}$$

où h_N est un hyperparamètre de *largeur de fenêtre* (qui dépend, en général, du nombre d'exemples employé), et V_N est un coefficient de normalisation§

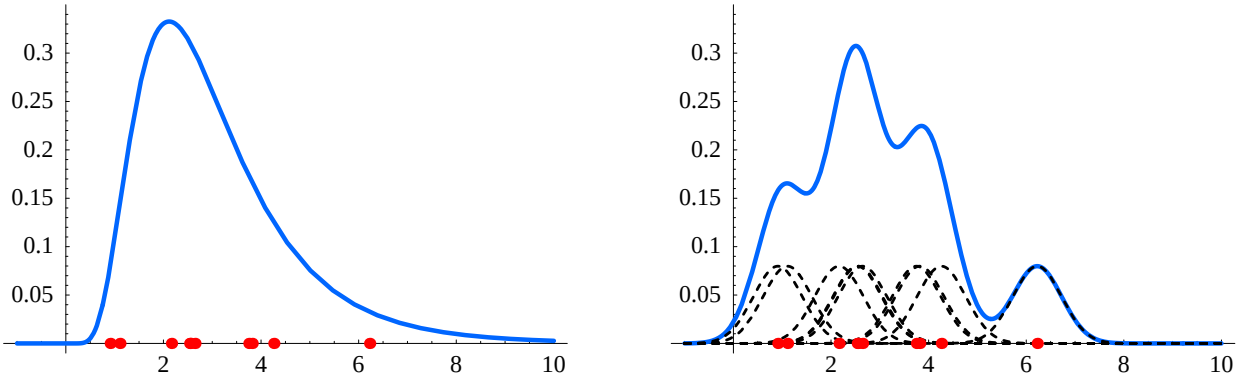
$$V_N = \int_{-\infty}^{\infty}\varphi\left(\frac{x}{h_N}\right)dx. \tag{2.18}$$

Dans beaucoup d'applications, les *noyaux gaussiens* s'avèrent entièrement adéquats,

$$\varphi(u) = \frac{1}{\sqrt{2\pi}}e^{-u^2/2}. \tag{2.19}$$

‡***Non-paramétrique**: Qui n'effectue aucune supposition quant à une forme fonctionnelle ou paramétrique particulière dont sont issues les données.*

§*Ici donné pour 1 dimension, mais la généralisation est triviale.*



▲ **Fig. 2.5.** *Estimation non-paramétrique de densité par fenêtre de Parzen.* ***Gauche*** *: distribution sous-jacente de laquelle on tire 10 points.* ***Droite*** *: on place un noyau gaussien (arbitrairement, $h_{10} = 0.5$ ici) sur chaque point ; la somme donne l'estimateur de densité.*

La figure 2.5 illustre cette estimation par un exemple simple de quelques points tirés d'une distribution log-normale. La largeur de fenêtre h_{10} est fixée arbitrairement dans cet exemple, mais elle peut être déterminée en pratique par des heuristiques motivées sur des bases asymptotiques (Scott 1992) ou, plus formellement, par validation croisée (Stone 1974).

On peut montrer que sous certaines conditions (voir par exemple Devroye, Györfi et Lugosi (1996)), l'estimateur $\hat{p}_X(x)$ est cohérent et converge* vers $p_X(x)$ lorsque $N \to \infty$.

*À divers degrés, dépendant des conditions imposées.

Extension des résultats théoriques

Les résultats bornant le sur- et le sous-entraînement de ADJ présentés aux propositions 2.1 et 2.2 ci-haut se généralisent assez aisément au cas étendu où un estimateur de la densité des entrées est utilisé plutôt qu'un ensemble non-étiquetté tiré de la véritable distribution. Les démonstrations complètes seront fournies dans un article de revue présentement en préparation.

2.4 Double-combo métrique régularisé : ADA

Récemment, une méthode de *régularisation adaptative*† employant les notions métriques décrites précédemment a été introduite par Schuurmans et Southey (2001). Cette méthode se veut plus robuste que celles de sélection de modèles car elle ne dépend pas d'une décomposition distrète d'une classe de fonctions maîtresse, et n'est pas sujette aux instabilités qui en découlent‡. La régularisation utilise ici les données non-étiquettées pour pé-

†*D'où provient le nom* ADA.

‡*Car les résultats de la sélection dépendent, en général, de la manière dont la classe est décomposée en sous-classes.*

naliser plus finement les fonctions individuelles selon un critère approprié.

L'essence du critère ADA est simple : plutôt que de chercher à optimiser uniquement le critère d'entraînement, il tente conjointement de trouver des fonctions qui se comportent de manière similaire à la fois sur les exemples d'entraînement *ainsi que dans leur voisinage**. À cet effet, on note les écarts entre une hypothèse et une fonction de référence $\phi(\cdot)$ qui sert « d'origine ». Pour les problèmes de régression, la fonction $\phi(\cdot)$ est généralement prise comme étant simplement la constante zéro ou encore la moyenne des valeurs cibles $\{y_i\}$ dans l'ensemble d'entraînement.

**En plus, bien sûr, de présenter une bonne performance sur le critère d'entraînement.*

Plusieurs formes de régularisation ont été comparées par SCHUURMANS et SOUTHEY (2001), et celle présentant les meilleurs résultats expérimentaux corrige la fonction objective originale par un terme multiplicatif, ce qui mène au problème d'optimisation suivant,

$$\min_{f \in \mathcal{F}} \hat{d}(f, P(Y|X)) \times \max\left(\frac{\tilde{d}(f,\phi)}{\hat{d}(f,\phi)}, \frac{\hat{d}(f,\phi)}{\tilde{d}(f,\phi)}\right), \tag{2.20}$$

où, comme ci-haut, $\hat{d}(f, P(Y|X))$ est le critère original d'entraînement, $\tilde{d}(f,\phi)$ est la distance entre f et ϕ estimée sur l'ensemble non-étiquetté, et $\hat{d}(f,\phi)$ est la distance estimée sur les entrées de l'ensemble d'entraînement.

Mettant de côté les difficultés d'optimisation numérique d'un tel critère[†], ADA tire son grand avantage de son adaptativité : la régularisation qu'il effectue est dépendante des particularités de l'ensemble d'entraînement, et n'est pas soumise à une pénalisation fixée. Ceci permet à l'algorithme d'agir plus agressivement pour prévenir le sur-apprentissage lorsque l'ensemble d'entraînement contient des « cas pathologiques », ce qu'aucun régulariseur fixe (même optimal) ne saurait faire, pour la même distribution d'entrées.

2.4.1 Contribution au domaine : lorsqu'on n'a pas de données non-étiquettées

L'extension de BENGIO et CHAPADOS (2001) décrite précédemment, qui utilise un estimateur non-paramétrique de la distribution des entrées au lieu d'un ensemble distinct de données non-étiquettées, s'applique également au critère ADA.

Les résultats expérimentaux obtenus dans le rapport précité, à la fois sur des problèmes synthétiques et réels, montrent que les versions étendues de TRI, ADJ, et ADA ne sont jamais statistiquement significativement pires

[†]Ces difficultés d'optimisation ne sont pas complètement négligeables ; en pratique, il est nécessaire d'effectuer plusieurs répétitions indépendantes d'une optimisation à base de gradient pour éviter de sombrer dans un mauvais minimum local. Nous avons aussi eu un certain succès avec des méthodes à barrières pour l'optimisation non linéaire (décrites plus bas) qui, en dépit de leur lenteur, semblent assez robustes aux minima locaux.

que les algorithmes originaux (ces derniers ayant néanmoins accès à de l'information supplémentaire sous forme d'un ensemble non-étiquetté), et sont parfois significativement meilleures*. De plus, la version étendue de ADA est, empiriquement, extrêmement robuste à la largeur de fenêtre de l'estimateur de Parzen, et n'a pas besoin d'ajustements minutieux pour présenter d'excellentes performances.

*La cause de cet avantage reste à élucider.

2.4.2 Contribution au domaine : optimisation non-linéaire sous contraintes

Une avenue possible pour mitiger les difficultés d'optimisation numérique présentées par l'éq. (2.20) est de résoudre un problème contraint en dimensionalité supérieure. Le problème original se reformule ainsi, en remplaçant le facteur $\max(\cdot,\cdot)$ par une variable contrainte θ,

$$\min_{f\in\mathcal{F}} \hat{d}(f, P(Y|X))\,\theta, \tag{2.21}$$

sous les contraintes

$$\begin{aligned} \hat{d}(f,\phi)\,\theta - \tilde{d}(f,\phi) &\geq 0, && (2.22)\\ \tilde{d}(f,\phi)\,\theta - \hat{d}(f,\phi) &\geq 0, && (2.23)\\ \theta &\geq 0. && (2.24) \end{aligned}$$

Ce problème peut dès lors se transformer en un problème non-contraint en incorporant à l'objectif principal des termes de « barrière » correspondant aux contraintes (FIACCO et MCCORMICK 1990; HILLIER et LIEBERMAN 2000), et admet une solution suivant une méthode d'optimisation non-linéaire classique (par ex. à base de gradient, *cf.* BERTSEKAS (2000)).

Quelques expériences préliminaires avec ce critère semblent indiquer qu'il est moins sensible aux minima locaux que le critère original, éq. (2.20), mais sa lenteur d'utilisation avec nos codes existants nous a poussé à remettre une considération détaillée à plus tard.

2.4.3 Travail futur

Les résultats de nos travaux les plus récents ont été publiés dans **?**) et **?**). Nous travaillons présentement à étendre les méthodes métriques utilisant des exemples non étiquettés (réels ou générés par un modèle de densité) au problème de sélection d'hyperparamètres pour la prévision de séries chronologiques non stationnaires, poursuivant certaines idées énoncées par BENGIO (2000).

3 Programmation dynamique approximative

Ce chapitre introduit une direction de recherche que nous entendons poursuivre au cours des prochains mois, et qui applique les algorithmes d'apprentissage à la gestion de portefeuille. Il se veut plus spéculatif en nature que les résultats présentés au chapitre précédent, et cherche à identifier des avenues prometteuses d'un point de vue théorique et pratique*.

*Et financier !

Nous établissons d'abord le cadre de gestion de portefeuille que nous considérons, et expliquons en quoi les approches classiques de valorisation de produits dérivés sont défaillantes pour les fins recherchées. Nous présentons ensuite un modèle d'investissement optimal basé sur la programmation dynamique, et terminons par une revue des méthodes d'approximation les plus prometteuses utilisant les algorithmes d'apprentissage.

Le lecteur trouvera pertinent de consulter un rapport CIRANO conjoint qui présente les résultats obtenus jusqu'à maintenant par l'équipe du laboratoire LISA sur la gestion de portefeuille d'options par algorithmes d'apprentissage.

3.1 Contexte : transiger options et autres produits dérivés

3.1.1 Rappel : sur les options et leurs stratégies

Les deux types fondamentaux d'options sont les *options d'achat* et les *options de vente* ; la première confère à son détenteur le **droit d'acheter** une *action sous-jacente*, à une date ultérieure déterminée (dite date de *maturité* ou *d'échéance*) pour un prix fixé d'avance (dit *prix d'exercice*). L'option de vente est similaire, mais confère le **droit de vendre**. Le *gain* d'une option (*payoff*) est la valeur de l'option à la date de maturité T. Soit S_T le prix du sous-jacent à cette date, et K le prix d'exercice de l'option. Le gain d'une option d'achat est donné par

$$\text{gain}_{\text{achat}}(S_T) = \max(S_T - K, 0).$$

De façon symétrique, le gain d'une option de vente est donné par

$$\text{gain}_{\text{vente}}(S_T) = \max(K - S_T, 0).$$

L'intérêt principal des options tient en cette non-linéarité dans la fonction de gain à maturité. Le *profit* d'une option exercée à maturité est égal au gain tel que ci-haut duquel on déduit le *prix d'achat initial* de l'option. On distingue une *position longue*, dans laquelle l'investisseur achète une option et avec elle les droits qu'elle confère, d'une *position courte*, dans laquelle l'investisseur vend une option et s'engage à honorer les obligations correspondantes envers l'acheteur. La figure 3.1 (a) et (b) illustre le profit d'une position longue dans une option d'achat et de vente respectivement. Le profit d'une position courte est la symétrique de ces courbes par rapport à l'abcisse.

Les *stratégies d'options* consistent à constituer un portefeuille formé de plusieurs options (généralement avec la même date d'échéance mais des prix d'exercice différents)*, et ainsi synthétiser une fonction de profit qui correspond aux perspectives de l'investisseur sur l'évolution du prix du sous-jacent. Plusieurs exemples classiques de ces stratégies sont illustrés à la figure 3.1 (c)–(h). Ces exemples tracent le profit de différents portefeuilles en fonction du prix final du sous-jacent.

*Sauf pour les « *calendar spreads* ».

3.1.2 Objectifs de la recherche proposée

Très peu de traités de « gestion moderne de portefeuille† » incluent les produits dérivés—tels que les options—comme véhicule de placement *bona fide*. Il est cependant bien connu (Hull 2000) que ces produits permettent de réaliser des stratégies de couverture (*hedging*) et de spéculation qui sont impossibles à répliquer facilement autrement‡.

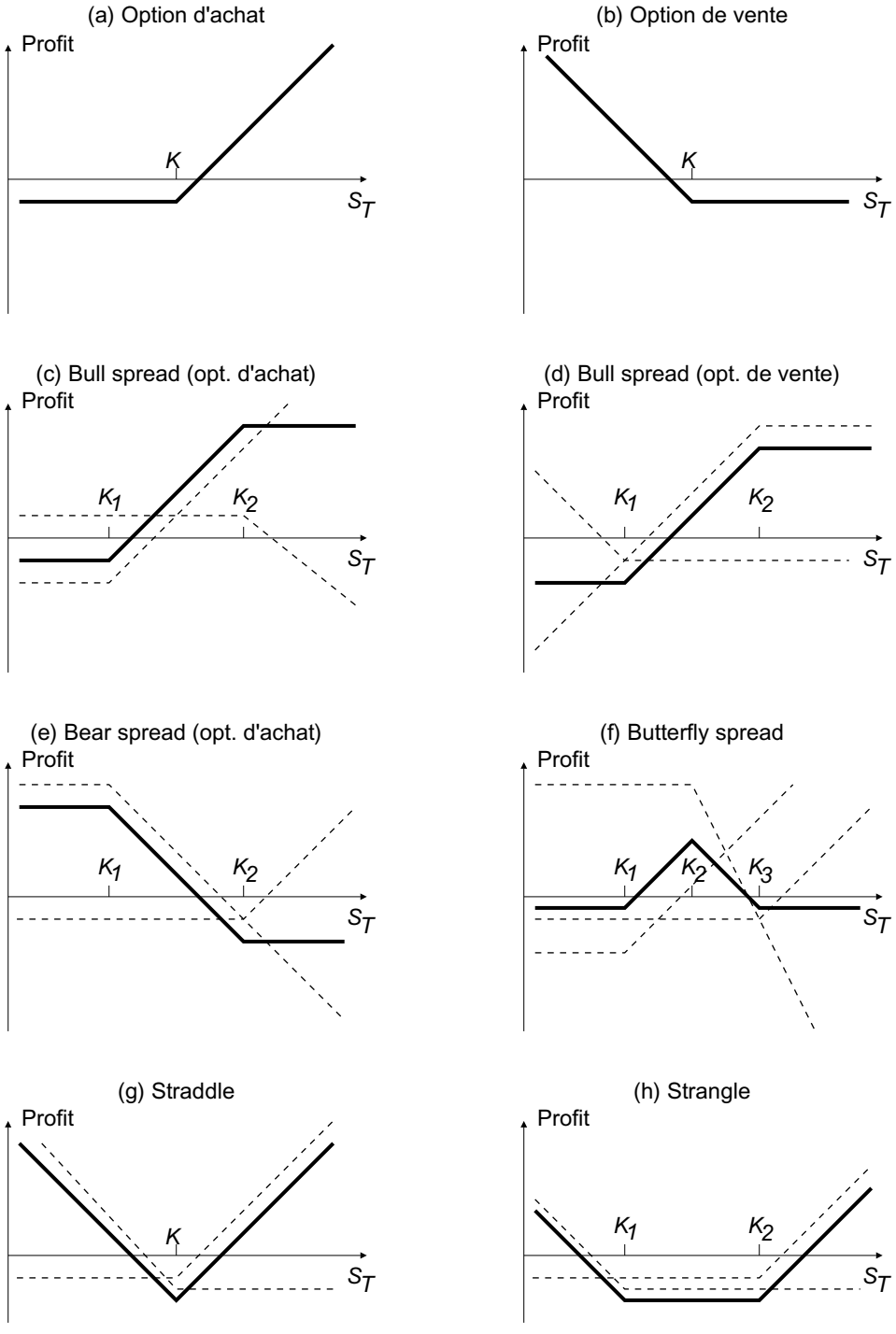
†*Que nous passons rapidement en revue en § 3.3.*

‡*À moins d'autoriser des transactions gratuites en temps continu pour* ***répliquer*** *le produit dérivé, ce que nous n'admettons pas, comme nous le détaillons plus bas.*

Depuis peu, certains auteurs commencent à considérer les produits dérivés du point de vue formel de leur profil risque–rendement (Korn et Wilmott 1998), afin de les traiter avec les outils de la théorie des portefeuilles dynamiques optimaux (Merton 1969; Pliska 1986; Duffie 1996; Korn 1997). Certains résultats ont été obtenus au sujet du problème du *portefeuille d'options optimal*§ en temps continu (Korn et Trautmann 1999; Korn et Korn 2001). Cependant, toutes ces approches reposent sur des suppositions fortes sur la dynamique des actifs et les propriétés des marchés financiers, suppositions que nous analysons et dont nous questionnons les fondements plus bas.

§*Ce problème est intéressant du point de vue théorique car il est un sur-ensemble strict du portefeuille d'actions optimal : toute action peut en effet être considérée comme une option d'achat avec un prix d'exercice de zéro.*

Au contraire, l'approche que nous proposons est fondée sur un ensemble d'hypothèses très réduit concernant le comportement des actifs financiers, essentiellement en se basant sur des réalisations passées (i.e. des données historiques) de trajectoires d'actifs. Nous mettons de l'avant un modèle de programmation dynamique qui offre certaines garanties d'optimalité quant



◀ **Fig. 3.1.** *Profil de profit net de différents portefeuilles d'options sur un même sous-jacent en fonction du prix du sous-jacent à maturité S_T. Ces portefeuilles sont créées par des positions longues ou courtes de plusieurs options simples.*

(a) Option d'achat simple.

(b) Option de vente simple.

(c–e) Spreads *permettant de spéculer sur une faible variation à la hausse ou à la baisse du sous-jacent.*

(f) Butterfly spread *permettant de spéculer sur une absence de variation du sous-jacent.*

(g–h) Straddle *et* Strangle *permettant de spéculer sur la volatilité (changement dans l'une ou l'autre direction).*

à la stratégie d'investissement à réaliser. Cependant, la version « naïve » du modèle (§ 3.4) souffrira de contraintes calculatoires reliées à la *malédiction de la dimensionalité* ainsi qu'à la caractérisation du comportement aléatoire des actifs financiers. Nous présentons des solutions (§ 3.5) basées sur les algorithmes d'apprentissage, que nous croyons efficaces dans l'aplanissement de ces cahots.

L'une des perspectives les plus excitantes de ce travail est la possibilité de « redécouvrir » les stratégies de la figure 3.1 entièrement par apprentissage statistique. Cette perspective, si elle se confirme, ouvre également la voie à des stratégies plus complexes mettant simultanément en oeuvre un nombre beaucoup plus grand de produits dérivés. Finalement, nous laissons entrevoir la possibilité (encore très spéculative) de valoriser les produits dérivés par un principe nouveau, celui de la *stratégie transactionnelle optimale*, par opposition au principe classique qui implique une stratégie de couverture ou de réplication.

3.2 Valorisation des options : la théorie classique

Afin d'établir un point de référence pour le lecteur, nous résumons ici brièvement les éléments fondamentaux qui sous-tendent la théorie classique de valorisation des options, qui sont les produits dérivés que nous souhaitons principalement considérer.

3.2.1 Le modèle de Black–Scholes

*Harrison *et* Kreps *(1979) introduisent une approche alternative de valorisation basée sur la* ***mesure martingale équivalente****. Voir, par exemple,* Hull *(2000) ou* Wilmott *(1998) pour une introduction au domaine*

Le modèle de Black–Scholes (Black et Scholes 1973; Merton 1973)* est à la base d'une théorie « rationnelle » de valorisation des options et autres produits dérivés : il se fonde sur l'idée que le juste prix d'une option devrait être le prix qu'il en coûte à un investisseur pour *répliquer avec un risque nul* le gain (*payoff*) de l'option.

Considérons le cas d'une option sur une action comme seul actif sous-jacent, et dont le prix est noté S. Nous effectuons l'hypothèse (trop forte) que le sous-jacent obéit à une marche aléatoire log-normale (mouvement brownien géométrique),

$$dS = \mu\, S\, dt + \sigma\, S\, dW, \tag{3.1}$$

où μ est le taux de rendement déterministe de l'actif (la « tendance » ou « drift »), σ est la volatilité, et $dW = \epsilon\sqrt{dt}$ avec $\epsilon \sim N(0,1)$ correspond à l'accroissement d'un processus de Wiener standard.

Sous cette hypothèse, et en supposant de plus que

1. l'investisseur peut éliminer le risque d'une position d'options en détenant une position opposée équivalente dans le sous-jacent*, et peut toujours maintenir cette position en rebalançant ce portefeuille en temps continu sans frais de transactions,
2. il n'y a aucune possibilité d'arbitrage sur les marchés financiers, c'est-à-dire qu'un investissement de risque nul *doit* avoir un rendement égal au taux sans risque en vigueur dans l'économie,

*Un principe appelé « *delta-hedging* ».

alors Black et Scholes ont montré que toute option, dont la valeur est notée $V(S,t)$ (une fonction du prix du sous-jacent S et du temps), doit obéir à l'équation différentielle partielle parabolique suivante,

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0. \tag{3.2}$$

Cette équation contient les mêmes paramètres que l'éq. (3.1), à l'exception de la tendance μ du sous-jacent qui est remplacée par le taux d'intérêt sans risque r. C'est l'une des caractéristiques remarquables du modèle de Black–Scholes que d'obtenir une solution pour le prix d'une option qui ne dépend pas de la tendance prévisible dans le prix du sous-jacent, mais uniquement du comportement imprévisible qui est amalgamé dans un paramètre de volatilité σ.

La solution de cette équation pour les conditions finales ou frontières appropriées donne la valeur de l'option pour tout S et t. Par exemple, pour une option d'achat européenne†, dont le gain à maturité T est donné par

$$V(S,T) = \max(S-K, 0), \tag{3.3}$$

où K est le *prix d'exercice*, nous obtenons la solution analytique de l'éq. (3.2)‡,

$$\begin{aligned} V(S,t) &= S\Phi(d_1) - Ke^{-r(T-t)}\Phi(d_2), \\ d_1 &= \frac{\log(S/K) + (r + \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}}, \\ d_2 &= d_1 - \sigma\sqrt{T-t}, \end{aligned} \tag{3.4}$$

où $\Phi(\cdot)$ est la fonction de répartition de la distribution normale centrée réduite.

†*Qui ne peut s'exercer qu'à maturité, au contraire d'une* ***américaine*** *qui peut s'exercer à tout moment jusqu'à maturité.*

‡*Par simplicité, nous omettons ici un éventuel flux de dividendes que verse le sous-jacent.*

Des formules analytiques s'obtiennent pour certains autres types de contrats « simples » ; autrement, on doit recourir à une variété de méthodes numériques pour résoudre l'éq. (3.2) (voir, par ex., Wilmott (1998) pour un aperçu de ce très vaste domaine).

3.2.2 Limites de Black–Scholes et extensions

Le modèle de Black–Scholes repose sur un ensemble d'hypothèses qui sont toutes plus ou moins fausses à divers degrés (Wilmott 1998). Il est

utile de passer ces suppositions rapidement en revue, et d'identifier dans quelle mesure elles affectent la valorisation.

Les sous-jacents suivent un mouvement Brownien géométrique Il s'agit de l'hypothèse que le sous-jacent évolue véritablement selon l'éq. (3.1). Bon nombre de résultats récents en économétrie financière rejettent (statistiquement) cette hypothèse à des degrés divers*.

*Voir Campbell, Lo et MacKinlay (1997) pour une revue de cette littérature.

Plusieurs extensions ont été proposées au mouvement brownien pour expliquer des phénomènes observés en pratique, tels que les « queues épaisses » de la distribution des rendements et les « sourires » (*smiles*) dans les courbes de volatilité implicite† ; nous notons ici simplement les modèles de diffusion avec saut (Merton 1976), les modèles à volatilité autorégressive conditionnelle ARCH/GARCH (Engle 1982; Bollerslev 1986), et les modèles à volatilité stochastique comme ceux de Hull et White (1987) et Heston (1993).

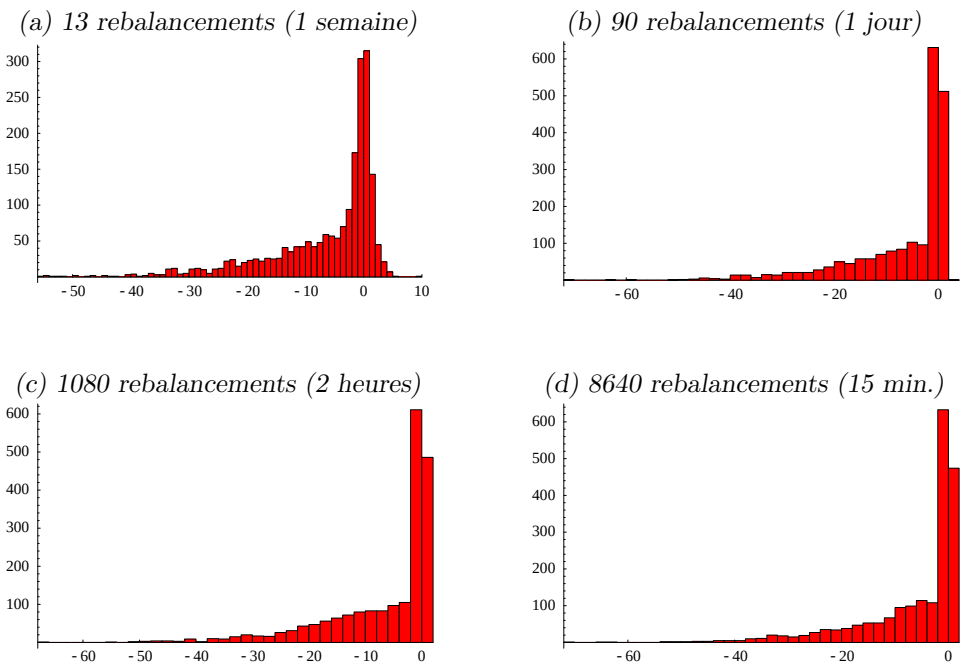
En dépit de leurs spécificités, tous ces modèles ont cependant à leur base une composante de diffusion similaire à celle du mouvement brownien géométrique. Nous pensons que même cette hypothèse peut être trop forte pour expliquer les rendements d'actifs financiers. Comme l'indique Campbell, Lo et MacKinlay (1997, p. 364) à propos de la modélisation par processus stochastiques en temps continu,

> « The specification of a continuous-time stochastic process (...) includes an infinity of parametric assumptions. Therefore, the convenience that continuous-time stochastic processes affords (...) should be weighted carefully against the many implicit assumptions that come with it. »

Rebalancement du portefeuille en temps continu Ceci est clairement impossible ; le portefeuille ne peut être rebalancé qu'en temps discret. L'effet du rebalancement en temps discret a été considéré par Boyle et Emanuel (1980) et Mercurio et Vorst (1997). Nous présentons quelques résultats de simulation à la figure 3.2 qui illustrent en termes simples l'impact *optimiste* de rebalancer en temps discret.

Nous constatons clairement que la distribution des profits excédentaires (c'est-à-dire en net du profit initial obtenu en vendant l'option) a une queue

†Le paramètre de volatilité σ est le seul qui ne soit pas observable dans le modèle de Black–Scholes. La *volatilité implicite* est la valeur de ce paramètre qui égalise les prix d'un modèle—par exemple Black–Scholes—avec ceux mesurés sur le marché. Lorsqu'on trace cette volatilité implicite en fonction du prix d'exercice d'une option, on obtient empiriquement une courbe qui ressemble à un sourire (ou une grimace), plutôt qu'une droite horizontale comme l'indiquerait le modèle de Black–Scholes. C'est l'un des paradoxes de ce modèle, et beaucoup de solutions ont été proposées pour y remédier à des degrés divers (Hull 2000; Wilmott 1998).



◀ **Fig. 3.2.** *Distribution des profits excédentaires obtenus en couvrant une option par rebalancements discrets (« delta hedging »), en supposant que l'actif suit l'éq. (3.1), sous la marche risque neutre ($\mu = r$). Chaque histogramme est le résultat de 2000 simulations de Monte Carlo, pour une option d'achat* at-the-money *au temps $t = 0$. Nous supposons que les marchés sont ouverts 24/7.*

négative très épaisse, et ce même en rebalançant très fréquemment et sous les hypothèses optimistes suivantes :

- Le sous-jacent suit un mouvement brownien géométrique.
- Le sous-jacent a le même taux de rendement que l'actif sans risque.
- Les marchés sont toujours ouverts et permettent de rebalancer en tout temps.

Le modèle de Black–Scholes est obtenu dans la limite d'une infinité de rebalancements, et devrait donner une distribution avec un pic à zéro. Cependant cette convergence est lente, et le rebalancement discret implique des risques substantiels.

Aucun frais de transaction Généralement, le niveau des frais de transaction dans un marché dicte la fréquence des rebalancements, ce qui a un impact direct sur le prix de l'option tel que démontré ci-haut. Le premier modèle de l'effet des frais de transaction pour des options standard* a été celui de Leland (1985), qui impose un rebalancement du portefeuille non pas en temps continu, mais à des intervalles distrets δt, où δt est un petit délai fixe et fini. Cette supposition résulte en un *ajustement de la volatilité* par un facteur proportionnel aux frais, en supposant une structure multiplicative pour ces derniers,

$$\text{frais} = \kappa \, |\nu| \, S,$$

**Dites « plain-vanilla », c'est-à-dire des options d'achat et de vente européennes.*

si ν parts sont achetées ou vendues au prix S, avec κ une constante.

L'analyse de Leland est étendue (entre autres) par Hoggard, Whalley et Wilmott (1994) à des options et portefeuilles arbitraires, et résulte en l'équation différentielle que doit satisfaire l'option sous une structure de frais multiplicative,

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} - \kappa\sigma S^2 \sqrt{\frac{2}{\pi\,\delta t}} \left|\frac{\partial^2 V}{\partial S^2}\right| + rS\frac{\partial V}{\partial S} - rV = 0. \tag{3.5}$$

Cette équation est l'un des premiers modèles *non-linéaires* en finance. L'effet de cette non-linéarité—commune à tous les modèles de frais—est fondamental : la somme de deux solutions de l'équation n'est pas nécessairement elle-même une solution. En d'autres termes, le prix d'une option ne peut être determiné indépendamment de la composition du reste du portefeuille ; la valeur du portefeuille *n'est pas égale à la somme* de la valeur de ses constituantes. Sous un modèle de frais de transaction, le prix d'une option ne s'établit pas de manière unique ; ce prix dépend du contexte (i.e. le portefeille) dans lequel l'option est transigée.

Il n'y a pas d'opportunités d'arbitrage Finalement, l'équation de Black–Scholes repose sur la « loi du prix unique », ou l'absence d'arbitrage, omniprésente dans toute la théorie financière moderne* : un investissement initial nul ne devrait pas générer de profit sans risque. En pratique, évidemment, de telles occasions existent, et certains acteurs des marchés financiers gagnent (généreusement) leur vie à les trouver.

*Souvent citée comme « *première loi de la thermodynamique financière* ».

Comme l'indiquent Lo et MacKinlay (1999, p. 16), « Market opportunities need not be market inefficiencies ». Il est entièrement concevable que la découverte de ces occasions d'arbitrage soient la juste récompense d'un risque couru† et non l'effet d'une irrationalité sans fondement théorique.

†*Par exemple, à chercher l'effet de « frictions » sur les marchés.*

En seconde analyse, cette hypothèse d'absence d'arbitrage est d'une ampleur difficile à quantifier : d'une part, les occasions d'arbitrage sont rares et rapidement corrigées par les marchés, et il est sage de ne pas compter sur elles pour fonder une théorie. D'autre part, précisément parce que toute la théorie de valorisation des produits dérivés repose sur l'*absence complète d'arbitrage*, tout écart de ces conditions aura des conséquences difficiles à anticiper.

3.2.3 Modèles non paramétriques

La complexité élevée des modèles récents de valorisation d'options, qui visent tous à palier aux déficiences du modèle de Black–Scholes a poussé certains chercheurs vers les *méthodes non-paramétriques* de valorisation. Ces modèles semblent tout indiqués pour les options, en permettant de capturer

des dépendances non-linéaires sans faire d'hypothèses paramétriques particulières.

Un premier pas dans cette direction est celui de Hutchinson, Lo et Poggio (1994), qui entraînent un réseau à fonctions de bases radiales* à apprendre la fonction de prix du modèle de Black–Scholes, sous l'hypothèse, bien sûr, que ce modèle est correct†. Des travaux similaires (Garcia et Gençay 2000) ont utilisé les prix du marché comme cibles de l'entraînement d'un réseau de neurones, mais ils sont tous basés sur l'hypothèse que le *marché valorise correctement*, autrement dit que les prix du marché réflètent adéquatement la valeur intrinsèque des titres.

Nous pensons, au contraire, que même cette hypothèse est trop forte : de plus en plus de résultats empiriques suggèrent que les marchés ne sont pas parfaitement *efficients*‡, et n'incorporent pas « instantanément » dans les prix toute l'information disponible à propos d'un titre (Campbell, Lo et MacKinlay 1997; Lo et MacKinlay 1999). En termes concrets, les marchés peuvent parfois être biaisés, inefficients, et victimes de bulles irrationnelles.

Il serait donc souhaitable de mettre au point et d'évaluer un modèle de valorisation ou de transaction non-paramétrique basé sur un minimum d'hypothèses, sans même compter sur la validité des prix observés sur le marché (ce modèle peut cependant se baser sur les prix observés comme référence contre laquelle transiger). C'est ce que nous tentons d'accomplir par le travail de recherche proposé.

** Qui sont similaires aux MLP du premier chapitre, mais avec un noyau gaussien plutôt que Tanh comme non-linéarité dans la couche cachée.*

† Le lecteur astucieux aura compris que c'est précisément ce qu'on cherche à éviter.

‡ Voir Samuelson (1965) pour une « preuve » classique de l'efficience des marchés.

3.3 Programmation dynamique et gestion de portefeuille

Le modèle « fondateur » de la gestion moderne de portefeuille est celui de Markowitz (1959), qui représente les actifs financiers par leurs rendements espérés ainsi que la covariance entre ces rendements§. L'investisseur rationnel, sous ce modèle, porte son attention sur les portefeuilles situés sur la *frontière efficiente*, constituée de ces portefeuilles dont le rendement est maximisé pour un niveau de risque donné. Il effectue ensuite son choix final en fonction de son niveau d'aversion au risque.

§ En les supposant gaussiens.

En dépit de sa simplicité conceptuelle et de sa facilité d'implantation, le modèle de Markowitz souffre du défaut d'être myope : il optimise les décisions au début d'une seule période, sans tenir compte de l'information qui devient disponible au cours de la période, ou du fait qu'un investisseur

prendra généralement des décisions sur un grand nombre de périodes au cours de sa vie.

Les modèles de programmation dynamique pour les choix de portefeuille visent tous à remédier à ces déficiences du modèle de Markowitz, et cherchent à optimiser globalement une *séquence de décisions d'allocation**. Ces modèles ont une longue histoire en finance, depuis les modèles en temps discret de Samuelson (1969) et Mossin (1968), jusqu'aux modèles en temps continu (sous le nom de *contrôle stochastique*) commençant avec Merton (1969). Duffie (1996), Korn (1997), et Korn et Korn (2001) fournissent un bon aperçu des résultats récent dans ce domaine.

**Ainsi généralement que les décisions de consommation au cours de l'horizon d'investissement.*

3.3.1 Contrôle stochastique optimal

Nous présentons ici un très bref sommaire de l'allocation basée sur le contrôle stochastique optimal (Krylov 1980; Bertsekas 2001), puisque cette forme se prête bien à une discrétisation et un traitement par programmation dynamique, ce que nous introduisons dans la section suivante. Le présent traitement est vaguement inspiré (avec simplifications) de Merton (1969) (voir aussi Merton (1990)). Il est à noter qu'une approche en temps continu très différente, dite par « martingales » (Pliska 1986), jouit d'une grande attention depuis quelques années en raison des simplifications analytiques considérables qu'elle autorise par rapport au contrôle stochastique optimal ; cependant, puisque notre but n'est pas principalement l'élégance symbolique, nous en restons avec la première approche, qui se veut d'esprit plus rapproché du modèle que nous présentons ci-bas†.

†*L'approche par martingales est aussi fondée sur un nombre plus grand d'hypothèses qui sont difficiles à outrepasser en vue d'atteindre un modèle non-paramétrique.*

Nous présentons ici un modèle simple de consommation et d'investissement, avec partage entre un seul actif risqué et l'actif sans risque. Soit $W(t)$ la richesse de l'individu au temps t, $c(t)$ son flux de consommation agrégé, et $\alpha(t)$ la proportion de sa richesse investie dans l'actif risqué. L'horizon d'investissement est $t \in [0, T]$. Nous supposons l'individu régi par une utilité de consommation $u(c(t), t)$, ainsi qu'une utilité de richesse terminale $B(W(T))$. L'investisseur cherche à prendre les décisions d'investissement et d'allocation qui maximisent son utilité totale espérée, ce qui mène naturellement au problème variationnel suivant, où $J(W(t), t)$ représente la valeur d'une richesse $W(t)$ au temps t,

$$J^*(W(0), 0) = \max_{c(t), \alpha(t)} E\left[\int_0^t u(c(t), t)\, dt + B(W(T))\right] \tag{3.6}$$

sous la *contrainte de budget* qui régit la variation instantanée $dW(t)$ de la valeur du portefeuille au temps t,

$$dW(t) = \underbrace{\{\alpha(t)(\mu - r)W(t) + rW(t) - c(t)\}dt}_{\text{Variation déterministe}} + \underbrace{\alpha(t)W(t)\sigma dz}_{\text{Variation stochastique}} \tag{3.7}$$

où dz est un incrément stochastique, par exemple issu d'un mouvement brownien, et avec la richesse initiale $W(0)$ donnée.

Une solution à ce problème est constituée de deux *fonctions*, $\alpha(t)$ et $c(t)$, et s'obtient en résolvant une équation de Hamilton–Jacobi–Bellman*.

*Voir, par exemple, Viala et Briys (1995), pour une dérivation simple.

3.3.2 Suppositions de la programmation dynamique

Le problème continu ci-haut se discrétise naturellement, et se présente alors aisément sous forme d'un problème de programmation dynamique, ce que nous abordons en détails dans la section suivante. Il est utile de garder à l'esprit les suppositions qui sous-tendent les « promesses d'optimalité » des solutions obtenues :

- L'investisseur est effectivement régi par l'utilité spécifiée, qui est indépendante d'une période à la suivante.
- Le modèle de la distribution du rendement des actifs† est adéquat. On ne peut mettre assez l'accent sur l'importance de ce dernier point. C'est là le frein principal à l'utilisation concrète de ces méthodes, et les méthodes d'approximation proposées en § 3.5 visent précisément à éviter une spécification à priori, mais permettre l'estimation (d'une fonction) de la distribution à partir de données historiques.
- Finalement, les modèles de programmation dynamique sont affligés de la diabolique *malédiction de la dimensionalité* (Bellman 1957), qui prévoit un accroissement exponentiel de la difficulté calculatoire avec le nombre de dimensions du problème. Les méthodes d'approximation que nous présentons sont conçues dans l'esprit de la mitiger.

†*Qui spécifie les probabilités de transition entre les états.*

3.4 Modèle de PD pour transiger des options

La présente formulation du problème de gestion optimale d'un portefeuille d'options est originale‡, bien qu'inspirée de modèles déjà proposés en finance (Samuelson 1969; Bertsekas 2001). Nous croyons qu'elle se prête bien aux méthodes d'approximation basées sur les algorithmes d'apprentissage, comme nous l'examinons plus bas.

‡*Au meilleur de notre connaissance.*

3.4.1 Définition de l'objectif d'investissement

Horizon considéré

Nous supposons dans ce qui suit un cadre d'investissement en temps discret sur horizon T fini, dans lequel, pour $t \in \{0, 1, \dots T\}$, *une période constante* (par exemple, une journée) s'écoule entre les instants t et $t + 1$§.

§*Nous sautons les week-ends et jours fériés.*

Dans le problème-type à résoudre, nous incluons dans le portefeuille toutes les options (de prix d'exercice différents) sur un seul titre sous-jacent* *venant à échéance à la même date*, et nous cherchons une stratégie optimale pour les transiger. Ces conditions permettent de limiter à un nombre fini les périodes à considérer†, et sont commodes pour éviter les complications reliées à la programmation dynamique sur horizon infini.

*Par exemple, l'indice S&P 500.

†Qui dépend de la durée pendant laquelle les contrats sont transigés, généralement quelques mois.

Actifs et portefeuilles

Définition 3.1 (Portefeuille) *Un portefeuille* $\mathbf{x}_t$ *défini par rapport à un ensemble d'actifs*‡ $\mathcal{A}_t = \{0, 1, \ldots, N\}$ *est le vecteur*

$$\mathbf{x}_t = (x_{t,0}, x_{t,1}, \ldots, x_{t,N})' \tag{3.8}$$

où $x_{t,0} \in \mathbb{R}$ *est la part détenue dans un* ***actif sans risque*** *(qui fait toujours partie de* $\mathcal{A}_t$*), et* $x_{t,1}, \ldots, x_{t,N} \in \mathbb{R}$ *sont un* ***nombre de parts*** *détenues dans les autres actifs (risqués), avec* N *le nombre d'actifs risqués dans le portefeuille.*

‡Cet ensemble est fixe, mais on peut sans trop de mal admettre des ensembles qui varient dans le temps, par exemple pour considérer des contrats qui viennent à échéance et les nouveaux qui commencent à se transiger.

Nous ne spécifions pas la structure des actifs autres que l'actif sans risque ; il peut s'agir d'actions, d'obligations, ou de produits dérivés (options, autres contrats à termes). Par simplicité de définition, nous admettons une position fractionnaire dans tous les actifs, que nous supposons infiniment divisibles§, ainsi que nous permettons les positions découvertes (i.e. négatives). De plus, nous supposons que tous ces actifs sont valorisés par rapport à une même devise.

§Nous verrons plus bas comment réconcilier cette supposition avec une réalité moins amicale.

Prix d'un actif Tout actif $i \in \mathcal{A}_t$ suit un processus de prix en temps discret $P_{t,i} \in \mathbb{R}^+$, qu'on peut assimiler au « véritable » prix de l'actif. Ce processus est observable étant donnée l'information disponible au temps t. L'actif s'achète au **prix d'achat** (*ask price*), dénoté $\overline{P}_{t,i}$, et se vend au **prix de vente** (*bid price*), $\underline{P}_{t,i}$, tous deux également observables étant donnée l'information au temps t. Nous imposons la condition

$$0 < \underline{P}_{t,i} \leq P_{t,i} \leq \overline{P}_{t,i}.$$

Le prix $P_{t,0}$ de l'actif sans risque est le prix d'une obligation dont le processus de prix s'accroit selon le taux sans risque en vigueur. Ce taux sans risque peut fluctuer à travers le temps, mais cette fluctuation est prise en compte dans le prix. Cette convention permet de tenir compte des questions d'actualisation du prix des autres actifs, et fournit une façon simple d'incorporer des taux « sans risque » stochastiques.

Dynamique du portefeuille

Nous supposons que l'investisseur doit liquider tous les actifs risqués avant l'atteinte du temps T, qui représente l'horizon d'investissement. En d'autres termes, le portefeuille final doit être constitué uniquement d'une position dans l'actif sans risque,

$$\mathbf{x}_T = (x_{T,0}, 0, 0, \ldots, 0)'. \tag{3.9}$$

Définition 3.2 *Une **politique d'investissement** π est une suite de fonctions $\{\mu_0, \mu_1, \ldots, \mu_{T-1}\}$, où $\mu_t : \mathbb{R}^{N+1} \mapsto \mathbb{R}^N$ est une **fonction de décision d'allocation** qui donne, depuis un portefeuille au temps t, le nombre de parts dans chaque actif risqué à acheter ou vendre à cet instant.*

Étant donné une décision $\mathbf{u}_t$ (qui peut être le résultat d'une politique d'investissement, i.e. $\mathbf{u}_t = \mu_t(\mathbf{x}_t)$), le portefeuille évolue selon la dynamique donnée par une fonction de transition $f(\mathbf{x}_t, \mathbf{u}_t)$ comme suit,

$$\begin{aligned} \mathbf{x}_{t+1} &= f(\mathbf{x}_t, \mathbf{u}_t) \\ &= \mathbf{x}_t + \begin{pmatrix} \Delta_{t,0} \\ \mu_t(\mathbf{x}_t) \end{pmatrix}, \end{aligned} \tag{3.10}$$

où $\Delta_{t,0}$ est la variation des parts de l'actif sans risque, constitué d'un terme correspondant à l'achat d'autres actifs, un terme correspondant à la vente d'actifs, et un dernier terme de frais de transactions,

$$\Delta_{t,0} = \frac{\text{vente}_t}{\overline{P}_{t,0}} - \frac{\text{achat}_t + \text{frais}_t}{\underline{P}_{t,0}}, \tag{3.11}$$

$$\text{vente}_t = \sum_{i=1}^{N} -\min\left(\mu_{t,i}(\mathbf{x}_t), 0\right) \underline{P}_{t,i}, \tag{3.12}$$

$$\text{achat}_t = \sum_{i=1}^{N} \max\left(\mu_{t,i}(\mathbf{x}_t), 0\right) \overline{P}_{t,i}, \tag{3.13}$$

$$\text{frais}_t = \sum_{i=1}^{N} \phi_{t,i}(\mu_{t,i}(\mathbf{x}_t)). \tag{3.14}$$

Nous admettons des frais de transactions $\phi_{t,i}$ qui dépendent de chaque catégorie d'actifs, ce qui est souvent le cas en pratique. Il sera fréquent de considérer des frais à *structure multiplicative*, où κ gouverne l'importance du frais,

$$\phi^{\mathrm{m}}(x) = \kappa \left| x \right|. \tag{3.15}$$

Définition 3.3 *Une politique π est **admissible** si elle n'engendre que des portefeuilles bornés, pour tout portefeuille initial borné,*

$$\|\mathbf{x}_0\| < \infty \overset{\pi}{\Longrightarrow} \|\mathbf{x}_t\| < \infty, \quad \forall t.$$

Utilité de l'investisseur

Nous considérons le cas où l'investisseur ne retire aucune utilité d'une consommation immédiate*, et cherche uniquement à maximiser l'utilité espérée de sa richesse terminale, ou en d'autres termes, l'héritage légué à la fin T de son existence (non stochastique). Soit Π l'ensemble des politiques admissibles. L'investisseur cherche à trouver une politique optimale π^* qui maximise une fonction d'utilité $U(\cdot)$ de la richesse terminale $x_{T,0}$,

$$\pi^* = \arg\max_{\pi\in\Pi} E\left[U(x_{T,0})\right]. \tag{3.16}$$

Nous supposons que la fonction d'utilité obéit aux conditions usuelles de la théorie financière†, c'est-à-dire que $U : (0,\infty) \mapsto \mathbb{R}$ est une fonction continue strictement concave et partout différentiable telle que

$$U'(0) \stackrel{\text{def}}{=} \lim_{x\downarrow 0} U'(x) = +\infty \tag{3.17}$$

$$U'(\infty) \stackrel{\text{def}}{=} \lim_{x\to\infty} U'(x) = 0. \tag{3.18}$$

Voici quelques exemples de fonctions d'utilité qui satisfont à ces conditions‡,

- **Utilité logarithmique :** $U(x) = \ln(x)$
- **Utilité puissance :** $U(x) = \frac{1}{\alpha}x^\alpha, \quad 0 < \alpha < 1.$

Ses besoins de consommation étant couverts autrement

†*Voir (*Viala *et* Briys *1995;* Korn *et* Korn *2001), par exemple.*

‡*Ces fonctions sont membres d'une classe plus générale, celle des fonctions à « aversion absolue pour le risque hyperbolique en la richesse » (*hara*), (*Viala *et* Briys *1995).*

3.4.2 Espace d'états et récurrences

La formulation par programmation dynamique (Bellman 1957; Bertsekas 2001) du problème (3.16) est la suivante. Nous introduisons successivement chaque ingrédient du modèle.

Structure de l'espace d'états

Afin de limiter l'explosion de la taille de l'espace d'états, nous supposons que les actifs ont simplement une dépendance de premier ordre entre eux, c'est-à-dire que la distribution des prix au temps $t+1$ dépend seulement de la distribution des prix au temps t ; il serait évidemment possible d'accomoder des dépendances d'ordre supérieur à un coût considérable en temps de calcul.

L'état au temps t, ξ_t, est formé de deux composantes, une qui est contrôlée et l'autre qui ne l'est pas :

1. L'état du portefeuille $\mathbf{x}_t$, c'est-à-dire le nombre de parts détenues dans chaque actif, tel que défini par l'éq. (3.8). C'est la composante contrôlée.
2. Le prix d'achat $\overline{P}_{t,i}$ et de vente $\underline{P}_{t,i}$ pour chaque actif, $i = 0,\ldots,N$. C'est la composante non contrôlée.

Nous avons donc,

$$\begin{aligned} \xi_t &= (x_{t,0}, \ldots, x_{t,N}, \overline{P}_{t,0}, \underline{P}_{t,0}, \ldots, \overline{P}_{t,N}, \underline{P}_{t,N})' \\ &= (\mathbf{x}_t', \mathbf{p}_t')'. \end{aligned} \tag{3.19}$$

Forme récursive de la fonction de valeur

Soit un état $\xi_t = (\mathbf{x}_t', \mathbf{p}_t')'$, et $U_t(\xi_t)$ l'ensemble des actions possibles au temps t dans cet état ; nous supposons une fonction de dynamique de portefeuille $f(\mathbf{x}_t, \mathbf{u}_t)$ telle qu'en l'éq. (3.10). Soit $P(\mathbf{p}_{t+1}|\mathbf{p}_t)$ la distribution jointe du prix des actifs au temps $t+1$, conditionnelle aux prix au temps t^*. Puisque nous supposons une utilité terminale seulement et une fonction de transition de la partie contrôlée $\mathbf{x}_t$ de l'état qui n'est pas stochastique, la *valeur optimale* $J_t^*(\xi_t)$ de l'état ξ_t au temps $t, 0 \leq t < T$, est donnée par une récurrence de Bellman particulièrement simple,

*Ce qui est une approximation quelque peu grossière ; $\mathbf{p}_{t+1}$ dépend probablement de bien plus que $\mathbf{p}_t$.

$$J_t^*(\mathbf{x}_t, \mathbf{p}_t) = \max_{\mathbf{u}_t \in U_t(\xi_t)} E_t \left[J_{t+1}^*(f(\mathbf{x}_t, \mathbf{u}_t), \mathbf{p}_{t+1}) \right] \tag{3.20}$$

$$= \max_{\mathbf{u}_t \in U_t(\xi_t)} \int J_{t+1}^*(f(\mathbf{x}_t, \mathbf{u}_t), \mathbf{p}_{t+1})\, dP(\mathbf{p}_{t+1}|\mathbf{p}_t), \tag{3.21}$$

où E_t dénote l'espérance conditionnelle à l'information disponible au temps t. On suppose de plus que le maximum existe dans cette équation (ce qui est toujours le cas si l'ensemble des contrôles $U_t(\xi_t)$ est fini), et que la distribution conditionnelle $P(\mathbf{p}_{t+1}|\mathbf{p}_t)$ est toujours telle que cette espérance existe.

La fonction de valeur terminale, qui termine la récursion ci-haut, est simplement donnée par l'utilité qu'on cherche à maximiser,

$$J_T^*(\mathbf{x}_t, \mathbf{p}_t) = U(x_{T,0}). \tag{3.22}$$

Si la distribution de la partie non contrôlée de l'état ne dépend que de la partie contrôlée, par exemple avec une forme $P(\mathbf{p}_t|\mathbf{x}_t)$, une astuce commune en programmation dynamique (Bertsekas et Tsitsiklis 1996, p. 48) permet de réduire considérablement l'espace d'états. Elle consiste en la définition de la *fonction de valeur simplifiée* suivante,

$$\tilde{J}(\mathbf{x}_t) = E[J(\mathbf{x}_t, \mathbf{p}_t)|\mathbf{x}_t], \tag{3.23}$$

et d'écrire la récurrence (3.20) en fonction de $\tilde{J}(\mathbf{x}_t)$; sous ces conditions, on peut montrer que la fonction de valeur simplifiée converge correctement. Malheureusement, dans le cas présent, cette technique est inapplicable à cause de la dynamique temporelle du prix des actifs ; en d'autres termes, étant donné seulement l'état du portefeuille, on ne peut spécifier la distribution des prix autrement que par la distribution inconditionnelle (ce qui est inutile).

3.4.3 Discussion sur la formulation

Cette formulation est onéreuse, car elle suppose un espace d'états à $3(N+1)$ dimensions, où $N+1$ est le nombre d'actifs. Or dans un système visant à transiger des options, N n'est pas petit. Considérons un actif sous-jacent typique, sur lequel on transige, pour une seule date d'échéance, des options avec 10 prix d'exercice différents. Pour chaque prix d'exercice, nous avons des options d'achat et de vente, ce qui fait un nombre d'actifs*

*1 obligation pour le taux sans risque, 1 prix du sous-jacent, 20 options.

$$N + 1 = 1 + 1 + 10 + 10 = 22,$$

pour une taille totale de l'espace d'états de $3 \cdot 22 = 66$ dimensions ! Il y a quelques façons de réduire la taille de cet espace, au prix d'un modèle moins réaliste :

- On peut inclure un seul prix $P_{t,i}$ pour les actifs, plutôt que séparément (le prix d'achat $\overline{P}_{t,i}$ et le prix de vente $\underline{P}_{t,i}$) ; on considère alors un écart (*spread*) constant (ou proportionnel au prix) entre le prix d'achat et de vente. Cette modification n'affecte pas le réalisme de manière catastrophique, et réduit la taille de l'espace à $2(N+1)$ dimensions.
- Plutôt que d'inclure séparément le prix des options d'achat et de vente, on peut n'inclure que les options d'achat (ou l'inverse), et déterminer le prix des options de vente à l'aide d'une relation de *parité* bien connue (Hull 2000). Ceci permet de réduire substantiellement le nombre N d'actifs, mais au prix d'une restriction de taille dans le réalisme : le modèle devient alors incapable d'exploiter toute opportunité d'arbitrage entre les prix des options de vente et d'achat.

De plus, l'intégrale (ou la sommation, si on discrétise la distribution) de l'éq. (3.20) doit s'effectuer en au moins $N+1$ dimensions, ce qui peut être coûteux, même avec des méthodes (quasi-) Monte Carlo ; de plus, elle suppose qu'on *possède un modèle* de la distribution jointe des rendements des actifs, et nous ne souhaitons poser aucune hypothèse sur ce modèle à priori.

Normalisation des prix

Un premier pas vers une formulation qui se prête bien aux algorithmes d'apprentissage est de normaliser adéquatement les prix des actifs. On peut écrire, pour un actif i,

$$P_{t,i} = P_{0,i} \frac{P_{1,i}}{P_{0,i}} \cdots \frac{P_{t,i}}{P_{t-1,i}}, \tag{3.24}$$

d'où

$$\log P_{t,i} - \log P_{0,i} = \sum_{\tau=1}^{t} \left(\log P_{\tau,i} - \log P_{\tau-1,i} \right). \tag{3.25}$$

On conserve dans l'espace d'états le « log-prix » normalisé, maintenu comme une somme d'incréments,

$$\tilde{p}_{t,i} \stackrel{\text{def}}{=} \log P_{t,i} - \log P_{0,i} \tag{3.26}$$

$$= \tilde{p}_{t-1,i} + \log P_{t,i} - \log P_{t-1,i}. \tag{3.27}$$

Cette variable a l'avantage d'être distribuée de manière approximativement symétrique autour de zéro, et est indépendante du prix initial de l'actif.

3.5 Méthodes d'approximation

Cette section couronne le travail préparatoire établi jusqu'à maintenant. Après un bref survol de la littérature concernant les utilisations des algorithmes d'apprentissage à la gestion de portefeuille, nous passons en revue les méthodes d'approximation usuelles basées sur la simulation. Nous élaborons ensuite le détail de méthodes d'approximation du problème (3.20) que nous proposons comme travail de recherche, et esquissons certains résultats théoriques que nous croyons être en mesure d'établir.

3.5.1 Algorithmes d'apprentissage et gestion de portefeuille

Les algorithmes d'apprentissage ont une longue histoire dans les applications financières, et particulièrement en gestion de portefeuille. Les premiers systèmes utilisant, par exemple, les réseaux de neurones, ont d'abord cherché à *prévoir* certaines quantités financières comme le prix ou la volatilité des actifs (Weigend et Gershenfeld 1993) afin de les utiliser dans un système d'allocation classique comme l'allocation moyenne–variance (Chapados 2000) ou employant des règles d'analyse technique (Kaufman 1998; Brock, Lakonishok et LeBaron 1992). Malheureusement, la plupart des séries financières sont affligées d'un niveau de bruit déconcertant qui rend la tâche de prévision au mieux ardue, et plus habituellement futile.

Plus récemment ont gagné en popularité une famille d'approches qui passent outre l'étape de la prévision pour *rendre directement des décisions d'allocation* (Bengio 1997; Moody et Wu 1997; Choey et Weigend 1997; Chapados et Bengio 2001a; Moody et Saffel 2001). Ces approches sont fondées sur l'observation empirique qu'il n'est généralement pas nécessaire de savoir faire de bonnes prévisions pour prendre de bonnes décisions : plutôt que d'optimiser un critère usuel de prévision tel que l'erreur quadratique, elles optimisent un critère de performance financière qui maximise le profit normalisé par une mesure de risque, tels que l'utilité quadratique classique (Markowitz 1959), le ratio de Sharpe (Sharpe 1966; Sharpe 1994), le

risque de perte (*downside risk*) (Sortino et van der Meer 1991), ou la valeur à risque du portefeuille (Jorion 1997).

Les architectures d'apprentissage le plus souvent utilisées dans ces contextes sont des réseaux de neurones à propagation avant (perceptrons multicouches) avec des entrées retardées (Rumelhart, Hinton et Williams 1986)* ou des réseaux récurrents entraînés par rétropropagation à travers le temps (l'article pré-cité, ou Werbos (1990))† et qui tentent de capturer une certaine dynamique temporelle.

Malheureusement, même si un réseau récurrent peut, en théorie avec un état interne suffisamment riche, apprendre la fonction de décision optimale qui maximise le critère de performance financière sur la séquence d'entraînement, le problème d'optimisation non linéaire‡ posé par l'apprentissage est formidable§. Les outils actuels d'optimisation numériques ne promettent guère mieux que l'atteinte d'un optimum local ou un plateau de la fonction objective. De plus, certains résultats théoriques indiquent que l'apprentissage de dépendances à long terme dans les réseaux récurrents est un problème ardu (Bengio, Simard et Frasconi 1994).

Ces aléas des réseaux récurrents nous font craindre qu'ils sont difficilement capables d'aller au-delà d'une représentation de l'épistructure d'une séquence temporelle ; ils nous encouragent d'autre part à considérer des formulations de programmation dynamique approximative qui laissent entrevoir de meilleures garanties d'optimalité globale.

*Qui généralisent les modèles de régression linéaire dont les entrées sont des moyennes mobiles.

†Qui généralisent les modèles linéaires de type arma.

‡Continu mais non convexe.

§Voir, par ex. Chapados et Bengio *(2001a) pour une formulation complète.*

3.5.2 Méthodes classiques d'approximation en PD

Les méthodes d'approximation de la programmation dynamique qui utilisent les algorithmes d'apprentissage sont aussi connues sous le nom *d'apprentissage par renforcement* ou *programmation neuro-dynamique* (Bertsekas et Tsitsiklis 1996; Sutton et Barto 1998) ; nous résumons ici les principales méthodes dans ce domaine, afin de les comparer avec celles que nous proposons dans la section suivante.

Afin de simplifier la présentation de cette section, nous considérons un problème de plus-court chemin stochastique sur horizon infini sans actualisation (Bertsekas 1995), sur un ensemble fini d'états notés $\mathcal{X} = \{1, 2, \ldots, n\}$, en plus d'un état terminal absorbant noté 0. Dans chaque état i, un choix de contrôle doit être effectué depuis un ensemble fini donné $U(i)$. Dans l'état i, le choix du contrôle u dicte la probabilité de transition à l'état j suivant, $p_{ij}(u)$. Une fois l'état terminal atteint, on ne peut le quitter car il est absorbant ; nous avons donc $p_{0j}(u) = 0, \forall u, \forall j \geq 1$. Nous supposons qu'au moment ou une transition est prise, un coût $g(i, u, j)$ est enregistré (stochastique, car fonction de l'état d'arrivée j) ; de plus, le coût à partir de l'état terminal est toujours zéro.

Nous considérons des politiques stationnaires $\mu(\cdot)$*, qui sont des fonctions donnant le contrôle à choisir ($\in U(i)$), depuis tout état i. Une fois la politique fixée, la séquence $i_0, i_1, \ldots$ des états traversés devient une chaîne de Markov, avec probabilités de transition,

*Nous omettons les détails techniques qui demandent des politiques dites « propres » qui induisent des fonctions de valeur bornées pour tous les états.

$$P(i_{k+1} = j | i_k = i) = p_{ij}(\mu(i)). \tag{3.28}$$

Soit N le nombre (stochastique) d'états traversés dans cette chaîne jusqu'à l'atteinte de l'état terminal 0. La *valeur* $J^\mu(i)$ d'un état i sous la politique stationnaire μ est le coût espéré accumulé à partir de cet état jusqu'à l'atteinte de l'état terminal,

$$J^\mu(i) = E\left[\sum_{k=0}^{N-1} g(i_k, \mu(i_k), i_{k+1}) \middle| i_0 = i\right]. \tag{3.29}$$

Nous trouvons également utile de définir le **Q-facteur** $Q^\mu(i,u)$, qui est le coût espéré sous la politique stationnaire μ sachant qu'on commence initialement dans l'état i en prenant l'action u, et en suivant par la suite la politique μ,

$$Q^\mu(i,u) = E\left[g(i_0, u_0, i_1) + \sum_{k=1}^{N-1} g(i_k, \mu(i_k), i_{k+1}) \middle| i_0 = i, u_0 = u\right]. \tag{3.30}$$

Pour des raisons évidentes, nous dénotons par J^* et Q^* les valeurs et les Q-facteurs sous la *politique optimale*.

Récurrences de Bellman La valeur de l'état i sous une politique fixée μ obéit à la récurrence de Bellman suivante

$$\begin{aligned} J(i) &= \sum_j p_{ij}(\mu(i))(g(i, \mu(i), j) + J(j)) && (3.31)\\ &= (T_\mu J)(i), && (3.32) \end{aligned}$$

où la notation $T_\mu J$, employée couramment en programmation dynamique, est définie par l'éq. (3.31) ; cet opérateur transforme une fonction de valeur J en la fonction de valeur au temps suivant sous la politique μ. De même, la valeur de l'état i sous la *politique optimale* μ^* obéit à la récurrence suivante

$$\begin{aligned} J^*(i) &= \max_{u \in U(i)} \sum_j p_{ij}(u)(g(i,u,j) + J^*(j)) && (3.33)\\ &= (TJ^*)(i), && (3.34) \end{aligned}$$

où, comme ci-haut, l'opérateur TJ représente la fonction de valeur au temps suivant sous la politique vorace découlant de J.

Itération des valeurs L'itération des valeurs est l'algorithme classique le plus simple pour résoudre la récurrence (3.33) ; il exploite la propriété de contraction de l'opérateur T pour mettre ainsi à jour le vecteur J (Bertsekas 1995),

$$J(i) \leftarrow \max_{u \in U(i)} \sum_j p_{ij}(u)(g(i,u,j) + J(j)). \tag{3.35}$$

On peut montrer que cette itération converge asymptotiquement vers J^*, à partir de toute fonction initiale J_0.

Itération des politiques C'est le second algorithme classique pour résoudre la récurrence (3.33). Partant d'une politique initiale μ_0, il procède en effectuant à chaque itération $k = 0, 1, \ldots$ les deux étapes suivantes :

1. **Évaluation de la politique** Il s'agit de trouver la fonction de valeur J^{μ_k} correspondant à la polique μ_k actuelle, telle que

$$T_{\mu_k} J^{\mu_k} = J^{\mu_k}. \tag{3.36}$$

*Et en supposant que μ_k est « propre ».

On peut pour ce faire itérer l'opérateur T_{μ_k}, puisque pour tout* J_0,

$$\lim_{i \to \infty} T^i_{\mu_k} J_0 = J^{\mu_k}.$$

†*En temps $O(n^3)$, où n est le nombre d'états.*

L'éq. (3.36) se résoud aussi directement comme un système linéaire† pour J^{μ_k}, en substituant la définition (3.31) de T_{μ_k}.

2. **Amélioration de la politique** À partir de J^{μ_k}, on calcule une nouvelle politique μ_{k+1} telle que

$$\mu_{k+1}(i) = \arg \max_{u \in U(i)} \sum_{j=0}^{n} p_{ij}(u)(g(i,u,j) + J^{\mu_k}(j)). \tag{3.37}$$

On itère jusqu'à ce que la politique améliorante μ_{k+1} demeure inchangée par rapport à μ_k. On peut montrer que le nombre d'itérations pour ce faire est fini (Bertsekas 1995), et généralement assez petit.

Formes d'approximation

Tout dépendant de nos connaissances du modèle, il existe trois façons fondamentalement différentes de prendre une décision (i.e. choisir un contrôle) à partir d'un état i, et les méthodes d'approximation en programmation dynamique peuvent intervenir dans chacun de ces trois cas.

Approximation de la fonction de valeur Si on possède la fonction de valeur optimale $J^*(i)$ de même qu'un modèle précis du système, c'est-à-dire qu'on connaît la distribution des coûts $g(i,u,j)$ d'une action u dans l'état i, ainsi que les probabilités de transition $p_{ij}(u)$, alors l'action optimale est trouvée en calculant

$$u^*(i) = \arg\max_{u \in U(i)} \sum_j p_{ij}(u)(g(i,u,j) + J^*(j)). \tag{3.38}$$

Pour cette situation, on peut opter pour une *approximation fonctionnelle de la fonction de valeur* $\tilde{J}^*$ paramétrisée par un vecteur θ, lequel minimise conceptuellement

$$\theta^* = \arg\min_{\theta} \sum_i w_i \|J^*(i) - \tilde{J}^*(i,\theta)\|^2, \tag{3.39}$$

avec w_i tel que $0 \leq w_i \leq 1, \sum_i w_i = 1$, est une pondération attribuée à l'état i, et $\|\cdot\|$ une norme donnée (par ex. la norme euclidienne). Cette fonction s'utilise alors de manière obvie pour trouver l'action optimale approximative,

$$\tilde{u}^*(i) = \arg\max_{u \in U(i)} \sum_j p_{ij}(u)(g(i,u,j) + \tilde{J}^*(j)). \tag{3.40}$$

Nous expliquons plus bas (§ 3.5.2 et suivantes) comment estimer directement le vecteur de paramètres θ sans connaître au préalable la fonction de valeur optimale, ou sans minimiser explicitement l'éq. (3.39), ce qui s'avère coûteux quand l'espace d'états est grand.

Approximation des Q-facteurs À défaut d'une connaissance précise du modèle, mais d'une connaissance des Q-facteurs, l'action optimale est donnée par

$$u^*(i) = \arg\max_{u \in U(i)} Q(i,u). \tag{3.41}$$

On peut alors approximer ces Q-facteurs par la méthode du *Q-Learning* (Watkins 1989; Watkins et Dayan 1992), dont nous omettons ici les détails.

Approximation directe de la politique Finalement, on peut tenter de procéder directement à l'apprentissage d'une politique, c'est-à-dire trouver une fonction $\tilde{\mu}(i;\theta)$ donnant immédiatement l'action à prendre pour l'état i,

$$\tilde{u}^*(i) = \tilde{\mu}(i;\theta) \tag{3.42}$$

Le vecteur de paramètres θ s'obtient conceptuellement en résolvant (en supposant des décisions réelles)

$$\theta^* = \arg\min_{\theta} \sum_i w_i \|u^*(i) - \tilde{\mu}(i;\theta)\|^2, \tag{3.43}$$

où l'action optimale $u^*(i)$ est donnée par l'éq. (3.38), et w_i est une pondération satisfaisant aux mêmes conditions que pour l'éq. (3.39).

Approximateurs linéaires

Une première architecture élémentaire d'approximation, dont on peut tracer les racines à Bellman et Dreyfus (1959), fait appel aux applications linéaires pour approximer la fonction de valeur. Cette architecture s'utilise ensuite conjointement à l'éq. (3.38) pour prendre des décisions. Il est usuel de l'utiliser conjointement à des *traits* (*features*) qui sont extraits de l'état courant.

Définition 3.4 (Trait) *Un* ***trait*** *(*feature*) est une fonction $\phi : \mathcal{X} \mapsto \mathbb{R}$ qui représente un aspect « significatif » d'un état $i \in \mathcal{X}$ sous la forme d'un nombre réel.*

Soit $\{\phi_1, \ldots, \phi_M\}$ un ensemble de traits. Étant donné un état i, nous définissons le *vecteur de traits* qui correspond à cet état,

$$\phi(i) = (\phi_1(i), \ldots, \phi_M(i))'. \tag{3.44}$$

Pour un problème particulier, les traits sont généralement choisis à la main selon les caractéristiques du problème et l'expérience du concepteur ; nous cherchons intuitivement un vecteur de traits qui, pour la majorité des états, résume adéquatement les propriétés essentielles de l'état aux fins d'en approximer la valeur associée sous une certaine politique.

L'approximateur linéaire, étant donné un vecteur de traits et de paramètres, estime la valeur de l'état i par

$$\tilde{J}(i;\theta) = \theta'\phi(i). \tag{3.45}$$

L'estimation du vecteur θ peut s'effectuer de différentes manières. Nous résumons brièvement les méthodes les plus courantes, soit l'itération de valeurs approximative, l'estimation par Monte Carlo, et les différences temporelles.

Itération des valeurs approximative

L'itération des valeurs approximative remonte, dans sa forme initiale, à Bellman et Dreyfus (1959). Partant d'une fonction de valeur approximative $\tilde{J}(\cdot;\theta_0)$ donnée par un vecteur de paramètres initial θ_0, elle sélectionne à l'itération k un sous-ensemble représentatif des états $S_k \subseteq \mathcal{X}$, et calcule une application de l'opérateur T pour tout $i \in S_k$,

$$\hat{J}_{k+1}(i) = \max_{u \in U(i)} \sum_j p_{ij}(u)(g(i,u,j) + \tilde{J}(j;\theta_k)). \tag{3.46}$$

Il faut ensuite calculer un nouveau vecteur de paramètres θ_{k+1} correspondant à $\hat{J}_{k+1}$, tel qu'un critère de coût quadratique (par exemple) est minimisé,

$$\theta_{k+1} = \arg\min_\theta \sum_{i \in S_k} w_i \|\hat{J}_{k+1}(i) - \tilde{J}(i;\theta)\|^2, \tag{3.47}$$

où, comme plus haut, w_i est une pondération attribuée à l'état i. On peut choisir l'ensemble d'états S_k depuis des connaissances à priori sur le problème, ou par simulation (tirages de Monte Carlo) de trajectoires dans l'espace d'états, sous une politique donnée, par exemple la politique vorace découlant de $\tilde{J}(i;\theta_k)$, ou une politique ϵ-vorace (Sutton et Barto 1998), qui choisit l'action vorace avec probabilité $1-\epsilon$, et une action aléatoire avec probabilité ϵ*.

** Les politiques ϵ-vorace permettent de balancer **l'exploration** et **l'exploitation**, qui est un problème classique en contrôle.*

Bornes de performance On peut établir certaines garanties de performance pour cet algorithme (Bertsekas et Tsitsiklis 1996). En particulier, nous supposons que l'architecture d'approximation est suffisamment flexible et que l'ensemble des états échantillonés est assez riche pour que

$$\|\tilde{J}_{k+1} - T\tilde{J}_k\|_\infty \leq \varepsilon \tag{3.48}$$

où $\tilde{J}_k \stackrel{\text{def}}{=} \tilde{J}(\cdot;\theta_k)$ est la fonction de coût anticipé après k itérations de l'algorithme, et $\|\cdot\|_\infty$ la norme sup. Dans ce cas, nous pouvons établir la borne suivante pour toute fonction de valeur initiale J_0,

$$\|T^{k+1}J_0 - J_{k+1}\|_\infty \leq k\varepsilon + \varepsilon. \tag{3.49}$$

De meilleurs résultats s'obtiennent pour des problèmes avec actualisation. Ces problèmes introduisent un facteur d'actualisation qui réduit la valeur présente d'un état futur. Soit α le facteur d'actualisation. On peut montrer que la convergence à une fonction « assez rapprochée » de la fonction de valeur optimale J^* est assurée, avec

$$J^* - \frac{\varepsilon}{1-\alpha}e \leq \lim_{k\to\infty}\inf J_k \leq \lim_{k\to\infty}\sup J_k \leq J^* + \frac{\varepsilon}{1-\alpha}e, \tag{3.50}$$

où e est un vecteur de 1 de la taille de l'espace d'états.

Ces bornes peuvent sembler quelque peu décevantes (linéaires en k pour les problèmes sans actualisation, et $O((1-\alpha)^{-1})$ pour les problèmes avec actualisation) ; cependant, le simple fait qu'elles existent indique que cet algorithme d'approximation est, dans son essence, bien fondé.

Évaluation de politiques par simulation pour des représentations tabulaires

Nous considérons maintenant une première de deux méthodes pour approximer l'étape d'**évaluation de la politique** de l'algorithme d'itération des politiques (p. 44). Plutôt que de résoudre un système linéaire de la taille de l'espace d'états pour évaluer la politique, nous l'évaluons approximativement en calculant le coût accumulé à partir de chaque état sous un grand nombre de *trajectoires simulées*. Pour ce faire, nous supposons qu'il est possible, depuis un état i et un contrôle u donnés, de *tirer un état j suivant* sous la distribution $p_{ij}(u)$, et d'obtenir le coût $g(i,u,j)$ résultant*. Partant d'un état initial i_0^m et sous une politique fixée μ, nous tirons un ensemble de trajectoires $(i_0^m, i_1^m, \ldots, i_N^m)$, où $i_N^m = 0$, et $m = 1, \ldots, K$ l'indice de trajectoire. Pour chaque trajectoire, nous calculons les coûts accumulés jusqu'à l'atteinte de l'état terminal,

$$c(i_0^m) = g(i_0^m, \mu(i_0^m), i_1^m) + \cdots + g(i_{N-1}^m, \mu(i_{N-1}^m), i_N^m). \tag{3.51}$$

La fonction de valeur estimée pour chaque état est simplement donnée par la moyenne échantillonale†,

$$\hat{J}(i) = \frac{1}{K} \sum_{m=1}^{K} c(i^m). \tag{3.52}$$

On peut également former un estimateur incrémental de J, qui est mis à jour après chaque trajectoire simulée,

$$\hat{J}(i) \leftarrow \hat{J}(i) + \gamma_m (c(i^m) - \hat{J}(i)), \tag{3.53}$$

où γ_m est un facteur de mise-à-jour qui peut être fonction de la trajectoire‡. On peut commencer la procédure avec $\hat{J}(i) = 0$.

Différences temporelles pour des représentations tabulaires

La méthode des différences temporelles est une généralisation de l'évaluation des politiques par tirages de Monte Carlo, introduite par Sutton (Sutton 1984; Sutton 1988). Elle se conçoit le plus aisément en réécrivant la mise-à-jour d'un état i_k de l'éq. (3.53) (pour une trajectoire donnée) sous la forme§

*Dans beaucoup de circonstances, il n'est pas nécessaire d'avoir une représentation explicite de $p_{ij}(u)$ pour ce faire, d'où l'intéret de cette technique.

†*Supposant ici une représentation explicite de chaque état de $\hat{J}$, sous forme d'un tableau.*

‡*Par exemple, $\gamma_m = \frac{1}{m}$; les conditions usuelles de l'approximation stochastique (*Benveniste, Metivier *et* Priouret *1990) doivent être remplies pour garantir la convergence de $\hat{J}(i)$ vers J^μ lorsque $m \to \infty$.*

§*En supposant toujours une représentation tabulaire de la fonction $\hat{J}$.*

$$\begin{aligned}\hat{J}(i_k) \leftarrow \hat{J}(i_k) + \gamma \Big(& \Big(g(i_k, \mu(i_k), i_{k+1}) + \hat{J}(i_{k+1}) - \hat{J}(i_k) \Big) \\ & + \Big(g(i_{k+1}, \mu(i_{k+1}), i_{k+2}) + \hat{J}(i_{k+2}) - \hat{J}(i_{k+1}) \Big) \\ & + \quad \cdots \\ & + \Big(g(i_{N-1}, \mu(i_{N-1}), i_N) + \hat{J}(i_N) - \hat{J}(i_{N-1}) \Big) \Big).\end{aligned}$$

Cette mise-à-jour s'abrège

$$\hat{J}(i_k) \leftarrow \hat{J}(i_k) + \gamma \left(d_k + d_{k+1} + \cdots + d_{N-1} \right), \tag{3.54}$$

où la *différence temporelle* d_k est donnée par

$$d_k \stackrel{\text{def}}{=} g(i_k, \mu(i_k), i_{k+1}) + \hat{J}(i_{k+1}) - \hat{J}(i_k). \tag{3.55}$$

De façon intuitive, cette différence calcule la distance entre l'estimateur du coût anticipé pour un état donné, $\hat{J}(i_k)$, et le coût estimé basé sur la trajectoire simulée, $g(i_k, \mu(i_k), i_{k+1}) + \hat{J}(i_{k+1})$. Elle indique de quelle amplitude on doit modifier l'estimation de la valeur actuelle.

Algorithme TD(λ) Cet algorithme attribue une pondération exponentiellement décroissante aux différences temporelles (que nous abrégeons TD, de l'acronyme anglais) dans la mise à jour de la valeur d'un état donné i_k. Cette pondération est fonction de la « distance » dans une trajectoire simulée entre l'état i_k et la TD d_m. Une règle de mise-à-jour possible prend la forme

$$\hat{J}(i_k) \leftarrow \hat{J}(i_k) + \gamma \sum_{m=k}^{\infty} \lambda^{m-k} d_m, \tag{3.56}$$

avec $d_m \equiv 0, m \geq N$, et γ un pas de gradient (qui peut être fonction de k).

Si $\lambda = 1$, nous retrouvons la méthode d'évaluation de la politique par Monte Carlo, vue précédemment.

Il existe littéralement des douzaines de variations de la règle (3.56). Certaines opèrent « en-ligne » et calculent l'impact d'une nouvelle différence temporelle sur tous les états passés rencontrés dans la trajectoire (ce qu'on appelle les *traces d'éligibilité*) ; d'autres limitent de façon dure l'impact temporel d'une TD ; etc. Un examen empirique de différentes variantes est présenté dans Sutton et Barto (1998).

Garanties de convergence On peut montrer que pour une représentation tabulaire de la fonction $\hat{J}$, presque toutes les variantes d'évaluation de politique par différences temporelles convergent vers la véritable fonction de valeur J^μ, pour les problèmes sur horizon fini ou infini, avec ou sans actualisation (Bertsekas et Tsitsiklis 1996).

Amélioration de la politique Étant donné un estimateur $\hat{J}$ de la fonction de valeur J^μ sous la politique μ, calculé par l'algorithme TD(λ), on peut calculer une nouvelle politique en appliquant l'étape habituelle d'amélioration de polique de l'algorithme d'itération des politiques (éq. (3.37)).

TD pour des représentations non tabulaires

L'algorithme des différences temporelles se prête aussi très bien à une approximation paramétrique de la fonction de valeurs, par exemple l'architecture linéaire vue précédemment, ou les réseaux de neurones à propagation avant dont le principe est énoncé au chapitre 1. La seule condition est de pouvoir calculer le *gradient* de la fonction de valeur à un état donné par rapport aux paramètres θ. Ce gradient se calcule trivialement pour l'architecture linéaire, et efficacement pour les réseaux de neurones grâce à l'algorithme de rétropropagation.

Dans une version « hors-ligne » de l'algorithme TD(λ), nous commençons par tirer une trajectoire simulée $i_0, \ldots, i_N$, et mettons ensuite à jour le vecteur de paramètres θ en effectuant un pas de montée de gradient pondéré par l'ensemble des différences temporelles,

$$\theta \leftarrow \theta + \gamma \sum_{m=0}^{N-1} \nabla \tilde{J}(i_m, \theta) \sum_{k=m}^{N-1} d_k \, \lambda^{k-m}. \tag{3.57}$$

Des versions « en-ligne » de l'algorithme existent aussi.

Garanties de convergence Les garanties de convergence de l'algorithme TD(λ) sont beaucoup plus faibles pour les architectures d'approximation que pour les représentations tabulaires. Des exemples de *divergence* sont connus pour TD(λ) avec des architectures non-linéaires, de même que pour TD(0) (incluant les architectures linéaires) si la trajectoire des états n'est pas échantillonnée sous la politique μ. Dans le cas général, les meilleures garanties théoriques peuvent être établies pour TD(1), même si c'est la version qui converge le plus lentement en pratique, à cause de la grande variance de l'estimation. Des résultats solides peuvent être établis pour les architectures linéaires sous certaines conditions (Tsitsiklis et Roy 1997; Tsitsiklis et Roy 2001). L'étude de la convergence de ces algorithmes pour des approximateurs généraux est encore un domaine de recherche actif.

3.5.3 Contribution proposée : apprentissage direct de la politique

Pour apprécier la nature de la contribution proposée, il importe de garder à l'esprit les particularités du modèle d'allocation d'actifs présenté en § 3.4,

ainsi que les simplifications qu'il admet : au contraire du cadre habituel de la programmation dynamique (et de l'apprentissage par renforcement), on n'accumule pas de récompense additive à chaque étape, mais une seule grosse récompense (l'utilité de la richesse terminale) à la toute fin (période T). Par ailleurs, la dynamique de la partie contrôlée de l'état (le nombre de parts de chaque actif) est entièrement déterministe, et la dynamique de la partie non contrôlée (les prix des actifs au temps t) est purement stochastique (qu'on suppose quand même markovienne) et aucunement influencée par le contrôle*.

*En supposant bien sûr que l'investisseur est trop petit pour avoir un impact significatif sur les prix du marché.

Désavantages d'apprendre la fonction de valeur

Nous émettons de plus la conjecture que l'apprentissage de la fonction de valeur optimale (approximative) est un problème probablement **trop difficile** que ce qui est requis pour prendre des décisions optimales†. Pour comprendre cette intuition, nous remarquons que l'espace des fonctions de valeur est beaucoup plus grand que celui des politiques : à chaque fonction de valeur J est associée une politique vorace μ qui lui correspond, telle que

$$T_\mu J = TJ.$$

Ceci implique que, pour une politique donnée μ, pour tout état i et contrôle $u \in U(i)$, nous avons l'inégalité‡

$$\sum_j p_{ij}(\mu(i))(g(i,\mu(i),j) + J(j)) \geq \sum_j p_{ij}(u)(g(i,u,j) + J(j)).$$

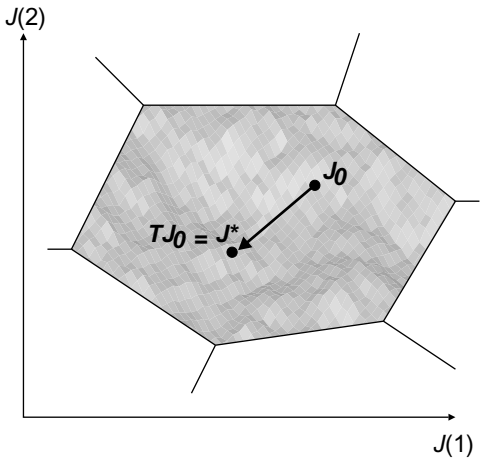
‡*Supposant ici une représentation tabulaire de la fonction de valeur J.*

L'ensemble des J qui satisfait à ce système d'inégalités forme un polyèdre convexe dans $\mathbb{R}^n$, qui est illustré schématiquement à la figure 3.3. À ce polyèdre correspond l'ensemble des fonctions de valeur qui donnent lieu à une politique vorace μ identique. Les différentes politiques possibles partitionnent $\mathbb{R}^n$ en un nombre dénombrable de cellules, alors qu'il y a une infinité non-dénombrable de fonctions de valeurs§. Cette caractéristique suggère que la fonction de valeur contient beaucoup plus d'informations que nécessaire pour prendre de bonnes décisions.

§*C'est la raison, en plus de la propriété de contraction de l'opérateur T, pour laquelle l'algorithme d'itération des politiques converge en un nombre fini d'itérations.*

Un deuxième désavantage de la représentation par fonction de valeur tient à la façon de prendre des décisions (*cf.* éq. (3.38)) : en plus de demander un modèle de la distribution de transition $p_{ij}(u)$, elle implique, pour chaque décision, une maximisation sur l'ensemble des actions possibles, ce qui est très coûteux dans un cadre de gestion de portefeuille, car le nombre d'actions possibles croît exponentiellement avec le nombre d'actifs. Ce désavantage se présente aussi avec une représentation des Q-facteurs (*cf.* éq. (3.41)).

†Il va sans dire que cette assertion dépend du problème ; l'impact d'une erreur sur J sera le plus significatif aux frontières des cellules illustrées en figure 3.3.



◀ **Fig. 3.3.** *Toutes les fonctions de valeur à l'intérieur de la cellule ombragée, par exemple J_0, impliquent une politique vorace identique, dans cet exemple la politique optimale.*

Finalement, la question du modèle est d'une importance capitale en pratique. Tel que mentionné à moult reprises dans ce chapitre, nous souhaitons éviter de poser un modèle *à priori* de la distribution des rendements des actifs ; demeure toujours la possibilité *d'estimer* un tel modèle de manière non-paramétrique ou semi-paramétrique (avec un réseau de neurones). Cependant, une telle approche introduit un élément de complication supplémentaire et probablement inutile : en plus d'apprendre la fonction de valeur (qui, comme nous l'avons suggéré, est trop de travail), il nous faut un modèle réaliste de la distribution conjointe conditionnelle du prix de tous les actifs, ce qui est non trivial pour un ensemble d'actifs constitué surtout d'options, car les corrélations entre les actifs possèdent une structure complexe.

Solution : approximer la politique

Ces inconvénients majeurs de l'apprentissage d'une fonction de valeur (ou dans une mesure similaire, des Q-facteurs) nous conduisent à **proposer l'apprentissage direct d'une politique**, c'est-à-dire de constuire une fonction qui retourne une approximation de l'action sous la politique optimale, étant donné un état ξ_t (nous reprenons ici la notation introduite en § 3.4),

$$\mathbf{u}_t = \tilde{\mu}_t^*(\xi_t) = f(\xi_t; \theta),$$

où f est une fonction calculée par une architecture d'approximation (par ex. un réseau de neurones) avec un vecteur donné de paramètres θ, qui demeure habituellement fixé dans le temps.

Apprentissage de la politique et synthèse d'un ensemble d'entraînement

Nous n'avons pas encore résolu le problème de l'apprentissage de la politique. Supposons pour l'instant que nous disposons d'un ensemble de paires d'entraînement contenant l'action optimale à prendre dans chaque état, $D = \{\langle \xi_t, \mathbf{u}_t^* \rangle\}$, pour un sous-ensemble représentatif étendu des états ξ_t qu'on pourra rencontrer en pratique.

L'apprentissage de la politique se réduit alors à un problème d'*apprentissage supervisé*, et nous pouvons utiliser les méthodes décrites au chapitre 1 pour le résoudre. Même si l'ensemble de données contient plusieurs millions d'exemples, les algorithmes actuels d'entraînement de réseaux de neurones (si c'est l'architecture qu'on retient) sont suffisamment efficaces pour s'acquitter de cette tâche dans des délais raisonnables.

Le véritable problème est donc celui de *trouver l'action optimale* $\mathbf{u}_t$ *pour chaque état* ξ_t que l'on souhaite considérer. Pour ce faire, la **clef que nous proposons** est de résoudre un autre problème de programmation dynamique, simplifié par rapport à celui introduit en § 3.4, en utilisant des *trajectoires historiques* d'évolution conjointe d'actifs. La méthode exacte envisagée se présente comme suit :

1. Considérons une trajectoire historique $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_T$ des prix d'un portefeuille d'actifs*. On suppose que cette trajectoire fait partie de notre ensemble d'entraînement, donc qu'elle est accessible en entier—on n'a pas à se soucier ici de contraintes de causalité.
2. **Étant donné** cette trajectoire complète, on trouve la séquence optimale des actions (discrétisées) à prendre à chaque étape pour maximiser l'utilité de la richesse terminale. Notons qu'il s'agit d'un problème de plus court chemin *déterministe*, car il a été supposé une réalisation complète (fixe) de la trajectoire. Ce problème de plus court chemin déterministe se prête à son tour naturellement à une solution par programmation dynamique classique (Bertsekas 2001) ou d'autres algorithmes de recherche heuristique. Nous examinons plus bas la complexité calculatoire de cette étape.
3. On recommence l'étape précédente avec toutes les trajectoires historiques disponibles dans la base de données (par ex. différentes actions et les options associées, des dates d'échéance différentes pour les options, etc.).
4. On *synthétise* un ensemble d'entraînement, à partir des résultats de l'étape précédente, formé de toutes les paires état/action optimale rencontrées sur la totalité des trajectoires historiques rencontrées ci-haut.
5. On estime une fonction qui approxime la politique optimale, par apprentissage supervisé depuis cet ensemble d'entraînement.

** On obtient ces trajectoires (car il en faudra plusieurs) à partir de bases de données historiques ; il est important pour notre approche d'utiliser de véritables données plutôt que des trajectoires résultant de simultations.*

Intuitivement, l'apprentissage « par coeur » de la séquence de décisions optimales sur *une seule* trajectoire réalisée aura peu de chances de bien généraliser, car la probabilité que cette trajectoire exacte se reproduise dans le futur est quasiment nulle. Cependant, nous envisageons l'hypothèse que la *moyenne empirique sur un grand nombre de trajectoires** présentera une bonne performance de généralisation. La vérification théorique (ou autrement empirique) de cette hypothèse est l'une des questions fondamentales que cette thèse se propose d'aborder, comme nous l'expliquons plus bas.

**Qui est apprise implicitement par le biais de l'apprentissage supervisé.*

Complexité calculatoire de l'étape simplifiée de programmation dynamique

L'étape « simplifiée » de programmation dynamique décrite plus haut cherche la meilleure séquence de décisions d'allocation sous une trajectoire (déterministe) réalisée de prix d'actifs. Il est instructif de contempler le coût de calcul de cette opération pour une situation « typique ». Pour le cas simple, effectuons les suppositions suivantes :

- L'horizon d'investissement habituel est de 6 mois, soit 125 jours ouvrables (si on souhaite transiger quotidiennement).
- Le portefeuille comporte 10 actifs.
- Le portefeuille peut assumer à tout moment une position longue, courte, ou neutre dans chaque actif. On limite pour l'instant la magnitude de la position à 1 part dans chaque actif.
- Les actions possibles sont d'acheter, vendre, ou ne rien faire, et ce pour chaque actif.

Donc, l'espace d'états compte 3^{10} états, et l'espace d'actions compte aussi 3^{10} actions. Pour appliquer l'algorithme de programmation dynamique, nous devons à chaque instant effectuer $3^{20} \approx 3.5 \times 10^9$ opérations. Pour un horizon d'investissement de 6 mois, nous atteignons le chiffre de 435×10^9 opérations. Cet ordre de grandeur est coûteux en temps de calcul sur les ordinateurs modernes, mais pas irréalisable.

Nous notons de plus que cette partie de l'algorithme demeure très sujette à la malédiction de la dimensionalité : une simple augmentation du nombre d'actifs, des positions admissibles dans le portefeuille, ou des actions possibles, porteraient ce problème bien au-delà des frontières du réalisable. Il est néanmoins utile de noter qu'un portefeuille constitué de seulement 10 titres est entièrement suffisant pour « redécouvrir » les stratégies classiques de transaction d'options mentionnées au début de ce chapitre.

Par ailleurs, nous croyons que des connaissances spécifiques au problème pourront éventuellemnt guider utilement le calcul d'une trajectoire optimale, en suggérant par exemple des heuristiques de recherche permettant d'éviter une considération exhaustive de toutes les paires états×actions. Ainsi, et c'est une voie de recherche ouverte, les méthodes d'optimisation combina-

toire heuristiques pourraient être mises à profit afin de traiter des problèmes de taille plus réaliste.

Questions de recherche

Par rapport à la méthode proposée, nous souhaitons nous pencher sur questions suivantes :

1. D'un point de vue théorique, est-ce que la politique apprise par un algorithme d'apprentissage sur un grand nombre de trajectoires converge vers la politique optimale ? Quel critère d'entraînement vaut-il mieux utiliser pour obtenir la meilleure performance de généralisation ?
2. Il est de même possible de vérifier empiriquement cette hypothèse dans un cadre plus restreint de simulation (pour lequel on pose un modèle précis et entièrement spécifié), et de comparer les politiques optimales approximatives apprises (par ex. par un réseau de neurones) à des politiques optimales découlant de l'application classique de la programmation dynamique.
3. Peut-on définir des heuristiques propres au cadre de gestion de portefeuille que nous considérons, et qui permettraient d'accélérer l'étape de programmation dynamique déterministe qui est centrale à l'approche proposée ? Quels algorithmes de plus-court-chemin sont les plus appropriés pour tirer parti de ces heuristiques*Il semble d'emblée évident que l'application complète de la programmation dynamique classique est trop coûteuse pour être viable à long terme. * ?
4. Finalement, comment se compare l'approche proposée à des méthodes classiques de gestion de portefeuille, utilisant par exemple un réseau récurrent entraîné sur un critère de performance financière ?

3.5.4 Autres applications : valorisation d'options

Pour conclure, nous contemplons une application alternative d'un modèle de décision tel que présenté ci-haut à la valorisation d'options. Rappelons que presque tous les modèles actuels de valorisation, tel que celui de Black–Scholes, sont basés sur une stratégie de *couverture* qui permet, sous des hypothèses très fortes, de trouver le prix unique que doit satisfaire l'option.

Cependant, comme nous l'avons vu, la notion de prix unique n'est plus applicable sous les modèles plus sophistiqués tenant compte des frais de transaction (*cf.* éq. (3.5)) : le prix d'une option dépend en fait de la composition entière du reste du portefeuille, ou autrement dit, on ne peut valoriser l'option indépendamment du contexte dans lequel elle est employée. De manière générale, ces modèles donnent une *bande de prix* à l'intérieur de laquelle doit se situer le prix de l'option.

Un modèle de décision peut s'utiliser comme principe alternatif de valorisation. Soit $\mathbf{u} = f(\mathbf{x}, \mathbf{p})$ l'action choisie par le modèle pour un état de portefeuille $\mathbf{x}$ et des prix d'actifs $\mathbf{p}$. Supposons qu'on fournit au modèle pour un portefeuille donné au temps t les *prix observés au marché* à ce temps, $\mathbf{p}_t$, et observons l'action recommandée $\mathbf{u}_t$ qui résulte. Trois conséquences sont possibles :

1. Le modèle recommande une action neutre sur tous les actifs, i.e. laisser le portefeuille inchangé. C'est signe que *sous le modèle*, les prix observés sur le marché tombent à l'intérieur de la bande de prix admissible.
2. Le modèle recommande une action d'achat sur certains actifs. Ceci indique que sous le modèle, ces actifs sont *sous-valorisés* par le marché. En d'autres termes, le modèle prévoit en espérance un gain d'utilité du portefeuille en augmentant la position dans ces actifs, pour une cause non spécifiée : ce peut être en tirant parti d'occasions d'arbitrage qui ont été détectées, ou par suite de la prévision d'une hausse imminente des prix. L'aspect important est que cette cause demeure *implicite*, étant encapsulée sous l'étiquette « stratégie transactionnelle optimale » que devrait suivre (en principe, sous les hypothèses de Markov, etc.) le modèle.
3. Le modèle recommande une action de vente sur certains actifs. De façon symétrique au point précédent, ceci indique que sous le modèle ces actifs sont *sur-valorisés* par le marché.

De façon duale, partant d'une recommandation d'achat ou de vente donnée par le modèle pour des prix du marché, on peut chercher quelle gamme de prix donnerait une recommandation neutre. Cette gamme de prix correspond alors à la bande admissible des prix que devraient avoir les actifs dans le portefeuille, sous une stratégie transactionnelle optimale. Cette recherche peut s'accomplir en faisant varier les prix donnés en entrée (utilisant, par exexple, les prix du marché comme point de départ), ou plus systématiquement par montée ou descente de gradient de la décision par rapport aux prix d'entrée (ce qui se calcule aisément si le modèle est un réseau de neurones).

Références

Akaike, H. (1973), « Information Theory and an Extension of the Maximum Likelihood Principle », *Second International Symposium on Information Theory*, p. 267–281.

Bellman, R. (1957), *Dynamic Programming*, NJ : Princeton University Press.

Bellman, R. et S. Dreyfus (1959), « Functional Approximation and Dynamic Programming », *Mathematical Tables and Other Aids to Computation 13*, p. 247–251.

Bengio, Y. (1997), « Training a Neural Network with a Financial Criterion Rather Than a Prediction Criterion », *Decision Technologies for Financial Engineering : Proceedings of the Fourth International Conference on Neural Networks in the Capital Markets (NNCM '96)*, World Scientific Publishing, p. 36–48.

Bengio, Y. (2000), « Gradient-Based Optimization of Hyper-Parameters », *Neural Computation 12*(8), p. 1889–1900.

Bengio, Y. et N. Chapados (2001, Juin), « Metric-Based Model Selection and Regularization in the Absence of Unlabeled Data », Rapport technique, Département d'informatique et de recherche opérationnelle, Université de Montréal, Montréal, Canada.

Bengio, Y. et D. Schuurmans (Édits.) (2002), *Machine Learning Special Issue on New Methods for Model Selection and Model Combination*. Morgan Kaufmann.

Bengio, Y., P. Simard et P. Frasconi (1994), « Learning Long-Term Dependencies with Gradient Descent is Difficult », *IEEE Transactions on Neural Networks 5*(2), p. 157–166, Special Issue on Recurrent Neural Networks, March 1994.

Benveniste, A., M. Metivier et P. Priouret (1990), *Adaptive Algorithms and Stochastic Approximations*. Springer.

Bertsekas, D. P. (1995), *Dynamic Programming and Optimal Control*, Volume II, Belmont, MA : Athena Scientific.

Bertsekas, D. P. (2000), *Nonlinear Programming* (Seconde ed.)., Belmont, MA : Athena Scientific.

Bertsekas, D. P. (2001), *Dynamic Programming and Optimal Control* (Seconde ed.), Volume I, Belmont, MA : Athena Scientific.

Bertsekas, D. P. et J. N. Tsitsiklis (1996), *Neuro-Dynamic Programming*, Belmont, MA : Athena Scientific.

Bishop, C. (1995), *Neural Networks for Pattern Recognition*, London, UK : Oxford University Press.

Black, F. et M. Scholes (1973), « The Pricing of Options and Corporate Liabilities », *Journal of Political Economy 81*, p. 637–659.

Bollerslev, T. (1986), « Generalized Autoregressive Conditional Heteroskedasticity », *Journal of Econometrics 31*, p. 307–327.

Boyle, P. et D. Emanuel (1980), « Discretely Adjusted Option Hedges », *Journal of Financial Economics 8*, p. 259–282.

Brock, W., J. Lakonishok et B. LeBaron (1992), « Simple Technical Trading Rules and the Stochastic Properties of Stock Returns », *Journal of Finance 47*, p. 1731–1763.

Campbell, J. Y., A. W. Lo et A. C. MacKinlay (1997), *The Econometrics of Financial Markets*. Princeton University Press.

Chapados, N. (2000), « Critères d'optimisation d'algorithmes d'apprentissage en gestion de portefeuille », Master's thesis, Département d'informatique et de recherche opérationnelle, Université de Montréal, Montréal, Canada.

Chapados, N. et Y. Bengio (2001a, Juillet), « Cost Functions and Model Combination for VaR-based Asset Allocation Using Neural Networks », *IEEE Transactions on Neural Networks 12*, p. 890–906.

Chapados, N. et Y. Bengio (2001b, Juillet), « Input Decay : Simple and Effective Soft Variable Selection », *Proceedings of the International Joint Conference on Neural Networks*, p. 1233–1237.

Choey, M. et A. S. Weigend (1997), « Nonlinear Trading Models Through Sharpe Ratio Maximization », *Decision Technologies for Financial Engineering : Proceedings of the Fourth International Conference on Neural Networks in the Capital Markets (NNCM '96)*, World Scientific Publishing, p. 3–22.

Devroye, L., L. Györfi et G. Lugosi (1996), *A Probabilistic Theory of Pattern Recognition*. Springer.

Duffie, D. (1996), *Dynamic Asset Pricing Theory* (Seconde ed.). Princeton University Press.

Dunis, C. et B. Zhou (Édits.) (1998), *Nonlinear Modelling of High-Frequency Financial Time Series*. John Wiley & Sons.

Efron, B. (1979), « Computers and the Theory of Statistics : Thinking the Unthinkable », *SIAM Review 21*, p. 460–480.

Engle, R. (1982), « Autoregressive Conditional Heteroscedasticity, with Estimates of the Variance of United Kingdom Inflation », *Econometrica 50*, p. 987–1007.

Fiacco, A. V. et G. P. McCormick (1990), *Nonlinear Programming : Sequential Unconstrained Minimization Techniques*, Classics in Applied Mathematics 4. Philadelphia, PA : Society for Industrial and Applied Mathematics.

Foster, D. et E. George (1994), « The Risk Inflation Criterion for Multiple Regression », *Annals of Statistics 22*, p. 1947–1975.

Garcia, R. et R. Gençay (2000), « Pricing and Hedging Derivative Securities with Neural Networks and a Homogeneity Hint », *Journal of Econometrics 94*, p. 93–115.

Geman, S., E. Bienenstock et R. Doursat (1992), « Neural Networks and the Bias/Variance Dilemma », *Neural Computation 4*(1), p. 1–58.

Hackl, P. (Édit.) (1989), *Statistical Analysis and Forecasting of Economic Structural Change*. Springer.

Harrison, M. et D. Kreps (1979), « Martingales and Arbitrage in Multiperiod Securities Markets », *Journal of Economic Theory 20*, p. 381–408.

Hassibi, B. et D. G. Stork (1993), « Second-Order Derivatives for Network Pruning : Optimal Brain Surgeon », *Advances in Neural Information Processing Systems*, San Mateo, CA, Morgan Kaufmann, p. 164–171.

Hastie, T., R. Tibshirani et J. Friedman (2001), *The Elements of Statistical Learning : Data Mining, Inference, and Prediction*. Springer.

Heston, S. (1993), « A Closed-Form Solution for Options with Stochastic Volatility with Application to Bond and Currency Options », *Review of Financial Studies 6*, p. 327–343.

Hillier, F. S. et G. J. Lieberman (2000), *Introduction to Operations Research* (Septième ed.). McGraw Hill.

Hinton, G. E. (1986), « Learning Distributed Representations of Concepts », *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, Hillsdale, NJ, Lawrence Erlbaum Associates, p. 1–12.

Hoerl, A. E. et R. Kennard (1970), « Ridge Regression : Biased Estimation for Nonorthogonal Problems », *Technometrics 12*, p. 55–67.

Hoggard, T., A. E. Whalley et P. Wilmott (1994), « Hedging Option Portfolios in the Presence of Transaction Costs », *Advances in Futures and Options Research 7*, p. 21–35.

Hornik, K., M. Stinchcombe et H. White (1989), « Multilayer Feedforward Networks Are Universal Approximators », *Neural Networks 2*, p. 359–366.

Hull, J. C. (2000), *Options, Futures and Other Derivatives* (Quatrième ed.). Prentice Hall.

Hull, J. C. et A. White (1987), « The Pricing of Options on Assets With Stochastic Volatility », *Journal of Finance 42*, p. 281–300.

Hutchinson, J. A., A. W. Lo et T. Poggio (1994), « A Nonparametric Approach to the Pricing and Hedging of Derivative Securities Via Learning Networks », *Journal of Finance 49*, p. 851–889.

Jorion, P. (1997), *Value at Risk : The New Benchmark for Controlling Market Risk*. Irwin McGraw-Hill.

Kaufman, P. J. (1998), *Trading Systems and Methods* (Troisième ed.). John Wiley & Sons.

Korn, R. (1997), *Optimal Portfolios*, Singapore : World Scientific Publishing.

Korn, R. et E. Korn (2001), *Option Pricing and Portfolio Optimization : Modern Methods of Financial Mathematics*, Volume 31 de *Graduate Studies in Mathematics*, Providence, RI : American Mathematical Society.

Korn, R. et S. Trautmann (1999), « Optimal Control of Option Portfolios », *OR-Spektrum 21*, p. 123–146.

Korn, R. et P. Wilmott (1998), « A General Framework for Hedging and Speculating with Options », *International Journal of Applied and Theoretical Finance 1*(4), p. 507–522.

Krylov, N. (1980), *Controlled Diffusion Processes*, Berlin : Springer.

Leland, H. E. (1985), « Option Pricing and Replication with Transaction Costs », *Journal of Finance 40*, p. 1283–1301.

Lo, A. W. et A. C. MacKinlay (1999), *A Non-Random Walk Down Wall Street*. Princeton University Press.

Markowitz, H. (1959), *Portfolio Selection : Efficient Diversification of Investment*. John Wiley & Sons.

Le Cun, Y., J. S. Denker et S. A. Solla (1990), « Optimal Brain Damage », *Advances in Neural Information Processing Systems*, San Mateo, CA, Morgan Kaufmann, p. 598–605.

Mercurio, F. et T. C. F. Vorst (1997), *Mathematics of Derivative Securities*, Chapter Options Pricing and Hedging in Discrete Time with Transaction Costs, p. 190–215. Cambridge University Press.

Merton, R. C. (1969), « Lifetime Portfolio Selection Under Uncertainty : the Continuous-Time Case », *Review of Economics and Statistics 51*(3), p. 247–257.

Merton, R. C. (1973), « Theory of Rational Option Pricing », *Bell Journal of Economics and Management Science 4*, p. 141–183.

Merton, R. C. (1976), « Option Pricing when Underlying Stock Returns are Discontinuous », *Journal of Financial Econometrics 3*, p. 125–144.

Merton, R. C. (1990), *Continuous-Time Finance*, Cambridge, MA : Blackwell Publishers.

Moody, J. et M. Saffel (2001), « Learning to Trade via Direct Reinforcement », *IEEE Transactions on Neural Networks 12*(4), p. 875–889.

Moody, J. et L. Wu (1997), « Optimization of Trading Systems and Portfolios », *Decision Technologies for Financial Engineering : Proceedings of the Fourth International Conference on Neural Networks in the Capital Markets (NNCM '96)*, World Scientific Publishing, p. 23–35.

Mossin, J. (1968), « Optimal Multiperiod Portfolio Policies », *Journal of Business 41*(2), p. 215–229.

Narendra, P. M. et K. Fukunaga (1977), « A Branch and Bound Algorithm for Feature Subset Selection », *IEEE Transactions on Computers 26*(9), p. 917–922.

Parzen, E. (1962), « On Estimation of a Probability Density Function and Mode », *Annals of Mathematical Statistics 33*(3), p. 1065–1076.

Pliska, S. R. (1986), « A Stochastic Calculus Model of Continuous Trading : Optimal Portfolios », *Mathematics of Operations Research 11*(2), p. 371–382.

Ripley, B. D. (1996), *Pattern Recognition and Neural Networks*. Cambridge University Press.

Rissanen, J. (1986), « Stochastic Complexity and Modeling », *Annals of Statistics 14*, p. 1080–1100.

Rumelhart, D. E., G. E. Hinton et R. J. Williams (1986), *Learning Internal Representations by Error Propagation*, Volume Parallel Distributed Processing : Explorations in the Microstructure of Cognition, Chapter 8, p. 310–362. MIT Press.

Samuelson, P. (1965), « Proof that Properly Anticipated Prices Fluctuate Randomly », *Industrial Management Review 6*, p. 41–49.

Samuelson, P. A. (1969), « Lifetime Portfolio Selection by Dynamic Stochastic Programming », *Review of Economics and Statistics 51*(3), p. 239–246.

Schuurmans, D. (1997, Juillet), « A New Metric-Based Approach to Model Selection », *Proceedings of the 14th National Conference on Artificial Intelligence*, Providence, RI,

Schuurmans, D. et F. Southey (2001), « Metric-Based Methods for Adaptive Model Selection and Regularization », *Machine Learning Special Issue on New Methods for Model Selection and Model Combination*, À paraître.

Scott, D. W. (1992), *Multivariate Density Estimation : Theory, Practice, and Vizualization*. John Wiley & Sons.

Sharpe, W. F. (1966, January), « Mutual Fund Performance », *Journal of Business*, p. 119–138.

Sharpe, W. F. (1994), « The Sharpe Ratio », *The Journal of Portfolio Management 21*(1), p. 49–58.

Sortino, F. A. et R. van der Meer (1991), « Downside Risk—Capturing What's at Stake in Investment Situations », *Journal of Portfolio Management 17*, p. 27–31.

Stone, M. (1974), « Cross-validatory choice and assessment of statistical predictions », *Journal of the Royal Statistical Society, B 36*(1), p. 111–147.

Sutton, R. S. (1984), *Temporal Credit Assignment in Reinforcement Learning*, Ph. D. thesis, University of Massachusetts, Amherst, MA.

Sutton, R. S. (1988), « Learning to Predict by the Method of Temporal Differences », *Machine Learning 3*, p. 9–44.

Sutton, R. S. et A. G. Barto (1998), *Reinforcement Learning : An Introduction*, Cambridge, MA : MIT Press.

Tibshirani, R. (1996), « Regression Shrinkage and Selection via the Lasso », *Journal of the Royal Statistical Society B 58*, p. 267–288.

Tsitsiklis, J. N. et B. V. Roy (1997), « An Analysis of Temporal-Difference Learning with Function Approximation », *IEEE Transactions on Automatic Control 42*, p. 674–690.

Tsitsiklis, J. N. et B. V. Roy (2001), « Regression Methods for Pricing Complex American-Style Options », *IEEE Transactions on Neural Networks 12*(4), p. 694–703.

Vapnik, V. (1998), *Statistical Learning Theory*. Wiley.

Viala, P. et E. Briys (1995), *Éléments de théorie financière*. Éditions Nathan.

Watkins, C. J. C. H. (1989), *Learning from Delayed Rewards*, Ph. D. thesis, Cambridge University, Cambridge, England.

Watkins, C. J. C. H. et P. Dayan (1992), « Q-Learning », *Machine Learning 8*, p. 279–292.

Weigend, A. et N. Gershenfeld (1993), *Time Series Prediction : Forecasting the future and understanding the past*. Addison-Wesley.

Weigend, A. S., D. E. Rumelhart et B. A. Huberman (1991), « Generalization by Weight-Elimination with Application to Forecasting », *Advances in Neural Information Processing Systems*, San Mateo, CA, Morgan Kaufmann, p. 875–882.

Werbos, P. J. (1990, October), « Back-Propagation Through Time :

What it Does and How to Do It », *Proceedings of the IEEE 78*(10), p. 1550–1560.

Wilmott, P. (1998), *Derivatives : The Theory and Practice of Financial Engineering*. John Wiley & Sons.