Université de Montréal

Prise de décision à partir de données séquentielles

par

François Gingras

Département de physique

Faculté des arts et des sciences

Thèse présentée à la Faculté des études supérieures

en vue de l'obtention du grade de

Philosophiae Doctor (Ph.D.)

en physique

janvier, 1999

Université de Montréal

Faculté des études supérieures

Cette thèse intitulée:

Prise de décision à partir de données séquentielles

présentée par

François Gingras

a été évaluée par un jury composé des personnes suivantes:

Jean Le Tourneux

(président du jury)

Bernard Goulard

(directeur)

Yoshua Bengio

(codirecteur)

René Garcia

(membre du jury)

Mario Marchand

(examinateur externe)

Serge Dubuc

(représentant du doyen)

Thèse acceptée le 13 octobre 1999

SOMMAIRE

Cette thèse propose des méthodes statistiques et des algorithmes pour la prise de décision à partir d'observations séquentielles. Nous proposons d'utiliser les erreurs de généralisation des modèles construits pour effectuer diverses tâches afin d'obtenir des estimés non biaisés de statistiques qui intéressent les utilisateurs de modèles pour résoudre des problèmes pratiques.

Nous présentons un algorithme qui permet, en utilisant les concepts de la théorie de l'apprentissage, de gérer les données manquantes. Cette situation peut survenir soit dans des problèmes statiques, où l'ordre des données n'est pas important, soit dans des situations où l'ordre des données importe. Nous montrons que cet algorithme peut permettre d'accroître la qualité des décisions concernant des problèmes où les données sont connues à différentes fréquences. Les valeurs manquantes ne sont pas nécessairement comblées par le meilleur estimé de la vraie valeur, mais plutôt par la valeur qui maximise le critère que l'utilisateur a choisi.

Nous proposons aussi un algorithme de type EM (Expectation-Maximization) s'appliquant à une mixture de mixtures de gaussiennes où certains paramètres sont partagés entre plusieurs gaussiennes. Le partage de paramètres est introduit ici en tenant compte des connaissances a priori que nous avons sur les données et afin de réduire l'espace des solutions possibles. Nous appliquons l'algorithme à un problème de reconnaissance et de classification d'états d'un réacteur nucléaire. L'algorithme s'applique notamment dans le cas où les étiquettes des états à classifier n'est pas strictement identifiées.

Nous étudions une statistique basée sur les erreurs de généralisation et nous construisons des tests d'hypothèses reposant sur cette statistique. Nous utilisons la

statistique hors-échantillon pour comparer les conclusions des résultats fournis par des méthodes traditionnelles (en utilisant tout l'échantillon) avec ceux fournis par notre approche. La statistique hors-échantillon nous permet de construire des tests qui nous permettent de distinguer entre deux hypothèses semblables, mais dont le rejet a des conséquences pratiques fort différentes.

Nous présentons et proposons des méthodes appropriées à cette statistique hors-échantillon qui reposent sur le "*bootstrap*" afin de faire des tests d'hypothèses. Nous étendons la notion de généralisation à des données non identiquement et indépendamment distribuées.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLES

REMERCIEMENTS

L'ensemble des travaux constituant cette thèse a été réalisé au laboratoire informatique des systèmes adaptifs (LISA) du département de recherche opérationelle et d'informatique de même qu'au groupe PhysNum, du département de physique et du CRM.

Je remercie vivement mes directeurs de recherches Yoshua Bengio et Bernard Goulard pour leur support scientifique et moral pendant toutes ces longues années. Je leur suis particulièrement reconnaissant de m'avoir encouragé à compléter cette thèse en dépit des embûches et tentations inhérentes au fait de poursuivre des études tout en accomplissant un travail exigeant à plein temps. Je remercie Yoshua Bengio de la confiance qu'il m'a témoignée en m'acceptant comme un de ses premiers étudiants alors que tout était à faire (et pour tout ce que nous avons fait en près de cinq ans).

Je remercie particulèrement Claude Nadeau pour l'intérêt qu'il a porté à la dernière partie de cette thèse. Je le remercie pour sa contribution et pour les diverses discussions allant de concepts généraux jusqu'aux infimes détails.

Je tiens à remercier Jean-Marc Lina pour sa collaboration tant sur les résultats apparaissant ici que pour ceux, nombreux, qui n'y apparaissent pas. Je lui suis reconnaissant pour les leçons privées qu'il m'a données lorsque je fréquentais plus assidûment le laboratoire de physique nucléaire.

Je remercie chaleureusement Joumana Ghosn pour les innombrables encouragements qu'elle m'a transmis chaque fois que les hésitations devant la poursuite de cette tâche regagnaient du terrain.

Je remercie messieurs Jean-Robert Derome et Raynald Laprade pour leur patience et leur compréhension.

Je remercie le FCAR et le CRSNG pour leur soutien financier.

Enfin, tous ceux qui, aux département de physique et au CRM, ont particulièrement formé mon esprit scientifique. Je pense à Jiri Patera, Yvan St-Aubin et Martin Leblanc.

Chapitre 1

INTRODUCTION

Cette thèse porte sur la prise de décision à partir de données séquentielles. Les algorithmes et les méthodes proposés sont appliqués à des séries financières ainsi qu'à des données provenant de senseurs installés dans un réacteur nucléaire. Le paradigme statistique sous lequel s'incrivent les articles regroupés ici est la théorie de l'apprentissage. La principale motivation de ces travaux repose sur l'hypothèse que dans certains cas, il n'est pas nécessaire d'utiliser la classe de modèles qui a généré les données pour prendre une décision, mais bien d'utiliser la classe de modèles qui généralisent le mieux. Nous disons de plus que, dans certains cas où la classe de fonctions où se trouve le vrai modèle est connue, nous devons utiliser un modèle choisi dans une autre classe de fonctions (plus simples) afin d'obtenir de meilleures performances de généralisation.

Les contributions de cette thèse sont d'ordres méthodologique et théorique. Nous proposons un algorithme permettant d'estimer les paramètres d'un réseau de neurones récurrents pour la prise de décision en présence de valeurs manquantes. Le modèle s'applique autant à des cas statiques qu'à des cas dynamiques. Dans ces dernières situations, des données disponibles à des fréquences plus basses ainsi que des données asynchrones sont traitées comme des valeurs manquantes.

Nous proposons aussi un algorithme EM (Expectation-Maximization) pour une mixture de mixtures de gaussiennes à paramètres partagés. L'algorithme EM est une procédure aidant à obtenir des estimateurs à maximum de vraissemblance pour certains problèmes où des données ne peuvent pas être observées, telle l'étiquette d'un

vecteur dans un problème de classification. Le partage de paramètres est un moyen de restreindre la capacité des modèles par l'incorporation de connaissances a priori sur les données. La capacité d'un algorithme d'apprentissage est une mesure de la "taille" de l'ensemble des fonctions qu'il peut réaliser. La vraie fonction, celle qui a généré les données, n'est peut-être pas dans cet ensemble de fonctions accessibles, ce qui implique la présence d'un biais dans l'erreur de généralisation. De plus, le choix particulier des données provenant de la distribution inconnue des données introduit une erreur appelée *variance* (car un autre ensemble de données tiré de la même distribution aurait conduit à une autre solution). De manière générale, l'augmentation de la capacité élargit l'ensemble des fonctions accessibles à l'algorithme d'apprentissage. Cela concourt à diminuer le biais, mais aussi à augmenter la variance de l'erreur de généralisation (Geman et al., 1992).

Utiliser des connaissances a priori est une façon de réduire le biais des transformations qui peuvent être appliquées sur les données par le modèle sans avoir à augmenter la variance, ce qui est d'autant plus nécessaire lorsque peu de données sont disponibles. Un avantage de l'algorithme proposé est qu'il s'applique dans les cas où les étiquettes des états à classifier ne sont pas strictement identifiées.

Enfin nous montrons comment une statistique, construite à partir des erreurs de généralisation, permet de tester l'hypothèse de prévision d'une série. Nous montrons que cette hypothèse ne doit pas être confondue avec l'hypothèse postulant l'existence d'une relation entre les variables utilisées pour prédire et la variable à prédire. Cette statistique hors-échantillon ne repose pas sur des hypothèses asymptotiques et est étudiée sur des échantillons de tailles finies. Nous étudions la puissance des tests d'hypothèses faits à l'aide de statistiques hors-échantillons. Nous appliquons la méthodologie que nous avons développée à la série des rendements de l'indice boursier canadien TSE300. Nous concluons qu'il existe possiblement, pour un horizon d'un an, une relation linéaire entre les rendements passés et les rendements futurs, mais que nous ne pouvons pas prévoir ces rendements avec la classe de modèles

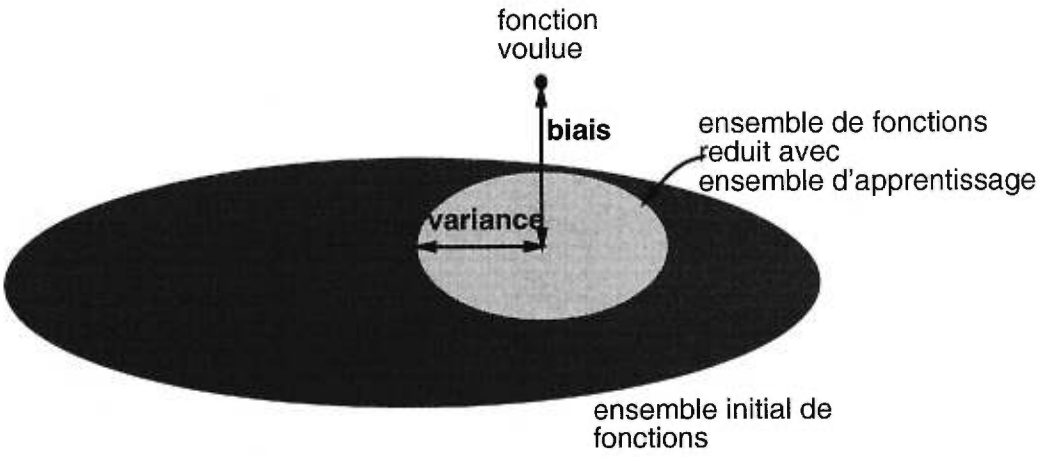

Figure 1.1. *Représentation de la décomposition de l'erreur de généralisation en composantes biais et variance. La grande zone ellipsoïdale foncée représente l'ensemble des fonctions accessibles que la machine d'apprentissage peut implanter. La taille de cet ensemble dépend de la capacité de la machine. La petite zone, en pâle, représente l'ensemble des fonctions qui est compatible avec l'ensemble d'apprentissage. L'erreur de généralisation peut-être décomposée en deux parties: le biais qui est une mesure de la distance entre la fonction désirée et l'ensemble initial; la variance qui dépend de la taille de l'ensemble réduit. La taille de l'ensemble réduit dépend du nombre d'exemples d'apprentissage.*

linéaires. Ces données ne sont pas tirées de façon indépendante et ne proviennent pas nécessairement de la même distribution (il est possible que les paramètres de la distribution changent dans le temps et que les données proviennent d'un modèle à plusieurs états). Nous ne faisons donc pas l'hypothèse que les données sont indépendamment et identiquement distribuées (i.i.d.) et nous étendons la notion de généralisation à des données non-i.i.d.

Nous pensons que ces contributions sauront intéresser les scientifiques, physiciens, mathématiciens et informaticiens qui sont de plus en plus sollicités pour résoudre des problèmes de prise de décision dans différents domaines de l'industrie.

1.1 Plan de la thèse

Cette thèse est constituée d'une série d'articles portant sur la prise de décision à partir de données séquentielles. Ce sont trois articles dont je suis un des co-auteurs. Chacun des articles présente un problème que l'on rencontre soit dans l'industrie nucléaire, soit en finance, plus précisément dans l'investissement boursier. Les articles ont été reformatés afin d'assurer une plus grande uniformité dans la présentation.

La suite de cette introduction expose des caratéristiques et des problèmes inhérents aux données séquentielles. On y présente aussi les concepts définissant la théorie de l'apprentissage qui sert de cadre théorique à cette thèse.

Le chapitre 2, consacré au premier article, présente un nouvel algorithme pour la prise de décision en présence de données manquantes. Nous avons tiré profit de la connectivité des réseaux de neurones récurrents et de l'apprentissage par des exemples de la relation qui nous intéresse pour mettre au point cet algorithme. Nous montrons qu'il s'applique à la prise de décision en présence de données manquantes dans les cas statiques et séquentiels.

Le chapitre 3, consacré au deuxième article, propose un algorithme permettant d'estimer efficacement les paramètres de mixture de gaussiennes, dont certaines parta-

gent des paramètres. Le modèle que nous proposons pour aider à la prise de décision des opérateurs de centrales nucléaires permet aussi de reconnaître des états atypiques pouvant se présenter pendant le fonctionnement des réacteurs.

Au chapitre 4, consacré au troisième article, nous présentons une statistique hors-échantillon que nous utilisons pour illustrer l'importante différence qu'il existe entre les hypothèses 1) pas de relation entre les entrées et les sorties et 2) pas de prévision possible à l'aide des entrées, étant donné une classe de fonctions. Nous proposons des méthodes pour estimer la variance de cette statistique hors-échantillon et nous comparons les résultats qu'elles fournissent sur des tests d'hypothèses avec ceux fournis par des statistiques conventionnelles. Nous présentons aussi une méthode de validation croisée séquentielle qui permet d'estimer l'erreur de généralisation de données non indépendantes.

En conclusion, nous indiquons quelques avenues de recherches et d'applications que nous entrevoyons dans l'immédiat.

1.2 Exposé de quelques problèmes

Les tâches auxquelles nous pouvons être confrontés lorsque nous devons prendre des décisions à partir d'observations temporelles présentent parfois quelques difficultés. Ces difficultés seront accrues si peu de données sont disponibles pour prendre une décision. Le processus, souvent inconnu, qui cause les phénomènes que nous mesurons, peut donner lieu à des observations non i.i.d., non stationnaires et non uniformément distribuées dans le temps.

Lorsque l'hypothèse i.i.d. n'est pas soutenable, plusieurs modèles et tests à notre disposition nécessitent des ajustements ou sont tellement biaisés qu'ils deviennent inappropriés. Dans le cas de la construction de modèles, ces ajustements peuvent prendre diverses formes. On peut tenter d'utiliser des modèles qui tiennent compte explicitement des dépendances temporelles, tels les réseaux de neurones récurrents

(Rumelhart et al., 1986a; Bishop, 1995; Hertz et al., 1991b) et les modèles de Markov cachés (Baker, 1975; Hamilton, 1989; Bengio and Frasconi, 1996). Cela peut aussi prendre la forme d'heuristiques qui exigent alors d'être évaluées empiriquement, ce qui peut être une complication supplémentaire si peu de données sont disponibles. Lorsque l'hypothèse d'indépendance est fausse, l'ordre des observations est important, et les méthodes pour évaluer l'erreur de généralisation doivent être modifiées.

La non-stationnarité, observée ou suspectée, des séries temporelles est une manifestation de données non-i.i.d. En présence de stationnarité, si nous estimons les paramètres d'une forme fonctionnelle posée a priori, l'augmentation du nombre d'observations se traduira par une convergence des paramètres estimés vers les paramètres de la distribution et une réduction de la variance des paramètres estimés, dans la mesure où nous supposons la bonne forme fonctionnelle. Cette assertion n'est pas pratique et ne constitue pas une définition opérationnelle de la stationnarité lorsque peu de données sont accessibles. Dans le cas de non-stationnarité, l'augmentation du nombre de données ne fera pas converger les paramètres estimés vers les paramètres de la distribution.

Enfin, l'observation de variables à des fréquences différentes ou irrégulièrement espacées dans le temps conduit à des complications supplémentaires dans le traitement de l'information pour la prise de décision. Si nous voulons intégrer les relations possibles qui existent entre les différentes variables disponibles, le modèle proposé doit pouvoir tenir compte des différentes échelles de temps et de la possibilité que la valeur d'une variable arbitraire manque parfois.

1.3 Théorie de l'apprentissage

La théorie de l'apprentissage concerne l'étude du problème qui consiste à minimiser une fonctionnelle de coût à partir d'un échantillon $z_1, z_2, \ldots, z_n$ indépendamment tiré d'une distribution inconnue $P(Z)$. En particulier, elle traite de l'inférence d'une

relation de dépendance à partir d'un nombre limité d'exemples de cette relation, et elle étudie la généralisation de cette relation sur de nouvelles observations provenant de $P(Z)$. Elle se distingue de l'inférence traditionnelle qui consiste à poser une forme fonctionnelle puis à en estimer les paramètres à partir des données, et où la plupart des résultats valent pour un grand ensemble de données.

Considérons une variable aléatoire $Z = (X, Y)$ ayant une densité de probabilité $P(Z)$ inconnue, et un ensemble $D_n = z_1, \ldots, z_n$ formé de n exemples tirés indépendamment de cette distribution. À titre d'illustration, considérons le cas où $X \in \mathcal{R}^n$ et $Y \in \mathcal{R}$ et soit $\mathcal{F}$ un sous-ensemble de l'ensemble des fonctions de $\mathcal{R}^n$ à $\mathcal{R}$. Nous nous donnons une fonction de coût Q qui caractérise la qualité d'une fonction f choisie dans $\mathcal{F}$ sur un Z particulier: $Q(f, Z)$ est une fonctionnelle de $\mathcal{F} \times \mathcal{R}^{n+1}$ à $\mathcal{R}$. L'objectif d'un algorithme d'apprentissage est de trouver une fonction $f \in \mathcal{F}$ qui minimise l'espérance du coût $Q(f, Z)$, c'est-à-dire l'erreur de généralisation de f:

$$G(f) = E_Z[Q(f, Z)] = \int Q(f, z)P(z)dz \tag{1.1}$$

Puisque la densité $P(z)$ est inconnue, nous ne pouvons pas minimiser $G(f)$. Par contre, nous pouvons mesurer et minimiser l'erreur empirique:

$$G_{emp}(f, D_n) = \frac{1}{n} \sum_{z_i \in D_n} Q(f, z_i) = \frac{1}{n} \sum_{i=1}^{n} Q(f, z_i) \tag{1.2}$$

Lorsque la fonction f est choisie indépendamment de l'ensemble d'apprentissage D_n, l'erreur empirique est un estimateur sans biais de l'erreur de généralisation. Un algorithme de minimisation du risque empirique est une procédure qui choisit la fonction $f \in \mathcal{F}$ telle que

$$f^* = \text{argmin}_{f \in \mathcal{F}} G_{emp}(f, D_n)$$

Différentes bornes sur la convergence des fonctionnelles de coût $G(f)$ et $G_{emp}(f)$ ont été dérivées, principalement par Vapnik (Vapnik, 1982). Ces bornes sont entre autres utiles pour contrôler la généralisation des systèmes d'apprentissage et

développer des méthodes applicables aux petits échantillons. Une formulation exprime la borne en fonction du nombre d'exemples d'apprentissage et une mesure de capacité appelée *dimension de Vapnik-Chervonenkis* (VC-dimension) de l'ensemble des fonctions $Q(f, z)$, $f \in \mathcal{F}$. Pour l'ensemble des fonctions affines la capacité de l'ensemble est simplement le nombre de paramètres libres. En général, la VC-dimension ne correspond pas aux nombres de paramètres.

La consistance de la théorie de l'apprentissage a aussi été étudiée (Vapnik, 1982; Devroye, 1988). Nous sommes intéressés à savoir comment l'erreur d'apprentissage approxime l'erreur de généralisation, notamment lorsque le nombre d'exemples tend vers l'infini. Sous certaines conditions (Vapnik, 1982; Vapnik, 1995), la plus grande différence entre $G_{emp}(f)$ et $G(f)$ tend vers zéro lorsque le nombre d'exemples tend vers l'infini. Entre autres, cela sera vrai si la capacité de la machine d'apprentissage est finie. De manière générale, l'erreur d'apprentissage et l'erreur de généralisation décroîtront avec l'augmentation initiale de la capacité. À partir d'une certaine capacité, pour un nombre donné d'exemples de la tâche à apprendre, l'erreur de généralisation cesse de décroître tandis que l'erreur faite sur les exemples d'apprentissage peut continuer de diminuer. À ce point, nous avons atteint la capacité optimale étant donné le nombre d'exemples et la classe des fonctions accessibles à la machine d'apprentissage. Ces concepts sont illustrés à la figure 1.2.

La manière la plus simple d'estimer l'erreur de généralisation revient à choisir un ensemble de test indépendant de l'ensemble d'apprentissage. Cela nous fournit une mesure non-biaisée de l'erreur, mais dont la variance peut être grande. Par contre, ce faisant nous nous trouvons à diminuer le nombre de données disponibles pour l'apprentissage. Lorsque les données sont peu nombreuses, nous nous trouvons à estimer l'erreur de généralisation d'un modèle construit avec très peu de données. Une façon d'obtenir un estimé non biaisé de l'erreur de généralisation sans devoir mettre de côté beaucoup de données utiles à la construction du modèle consiste à utiliser la validation croisée. Appliquer cette méthode revient à diviser aléatoirement

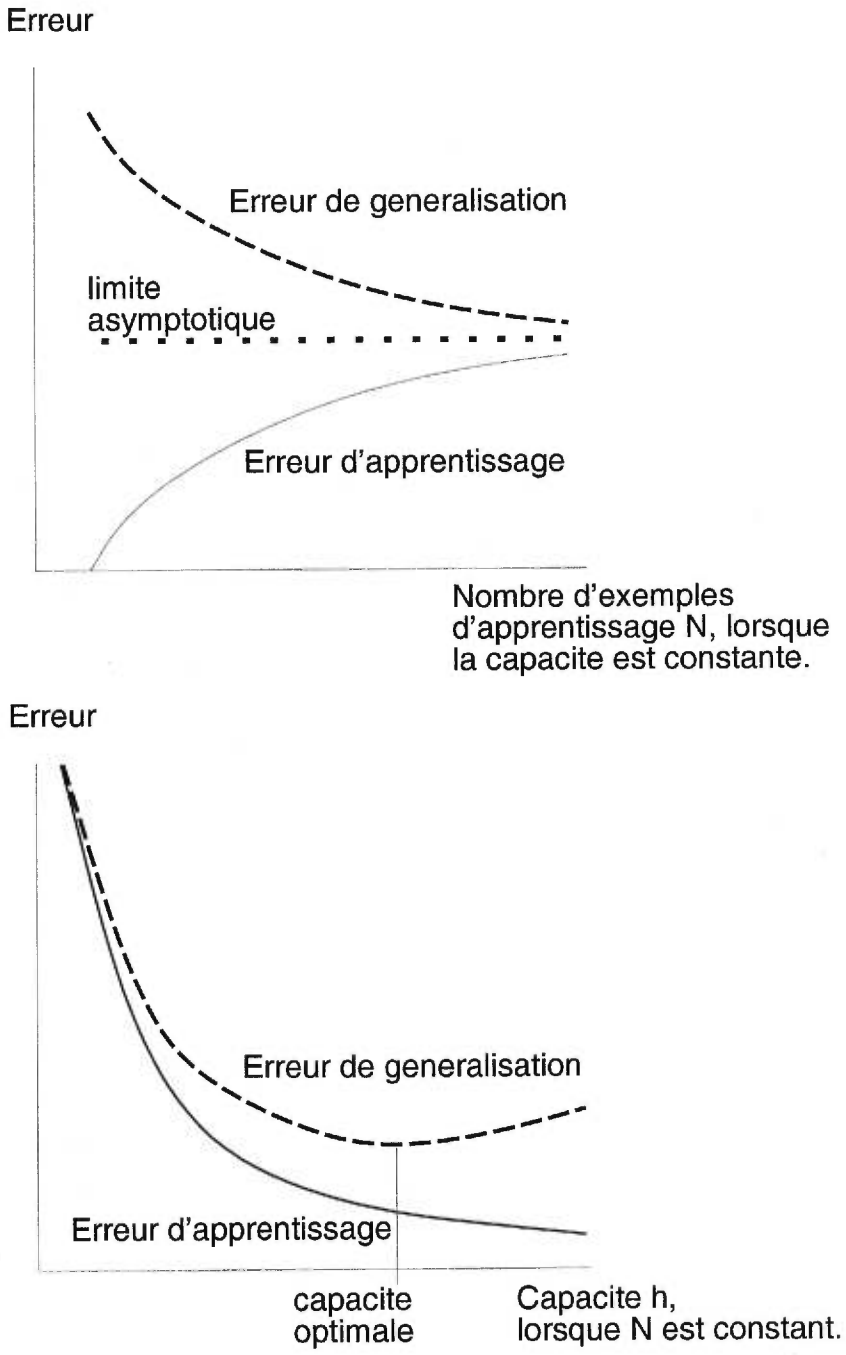

Figure 1.2. *Illustration de la courbe de l'erreur de généralisation* **en haut:** *pour une capacité donnée, accroître le nombre d'exemples d'apprentissage améliore la généralisation, mais augmente l'erreur d'apprentissage.* **en bas:** *pour un nombre d'exemples d'apprentissage donné, une valeur optimale de la capacité existe.*

l'ensemble des données en V sous-ensembles, puis à estimer l'erreur de généralisation sur un des sous-ensembles en utilisant les (V-1) autres sous-ensembles pour construire le modèle. Nous pouvons ainsi obtenir V estimés de l'erreur de généralisation et en faire la moyenne. Cette méthode est valide seulement pour des données i.i.d.

Chapitre 2

PREMIER ARTICLE: UNE FAÇON DE GÉRER LES VALEURS MANQUANTES

Dans ce chapitre, nous proposons un algorithme pour entraîner un réseau de neurones récurrent de manière à pouvoir gérer une situation impliquant des valeurs manquantes. L'occurrence d'une variable manquante peut être aléatoire et les situations où des variables manquantes peuvent se présenter sont nombreuses dans les problèmes pratiques. Les données peuvent provenir de senseurs suceptibles de faire défaut, l'information attendue à un certain moment se laisse attendre, des personnes peuvent refuser de répondre à certaines questions de sondage, certains tests médicaux ne peuvent pas être menés, etc. Nous considérons aussi le cas où des séries temporelles fournissent de l'information à des fréquences différentes, comme cela se produit lors de la publication de données économiques mensuelles ou trimestrielles ajoutant de l'information à des séries quotidiennes.

La façon la plus simple de traiter les valeurs manquantes est d'omettre les vecteurs qui ont des composantes manquantes. Si peu de données sont disponibles, cette approche amenuise l'espoir de trouver un modèle pour les données. Une alternative à cette approche triviale consiste à construire un modèle, souvent linéaire, entre une composante des vecteurs, disons la première si elle présente un ou plusieurs cas de valeurs manquantes, et les autres variables des vecteurs. Nous pouvons alors utiliser la moyenne conditionnelle fournie par ce modèle pour estimer les valeurs manquantes. Cette méthode est un raffinement de celle qui consiste à utiliser la moyenne inconditionnelle des composantes pour remplir les valeurs manquantes. Ces estimés de moyennes sont ensuite, par exemple, utilisés pour estimer la matrice de covariance.

Dans certaines situations, nous possédons peu de détails sur le modèle générateur des données observées. Mais notre but ici demeure la construction d'un modèle qui nous donnera des prédictions ou des décisions acceptables. Ce but en tête, nous disons qu'il importe davantange de combler les valeurs manquantes par des valeurs qui nous aident d'abord à atteindre cet objectif, plutôt que de construire un modèle où des ressources sont allouées à la recherche de la valeur la plus proche possible de la valeur manquante. Dans un espace de variables d'entrées de grandes dimensions, chercher un modèle qui nous donne la meilleure valeur pour la valeur manquante peut exiger l'estimation d'un très grand nombre de paramètres, par exemple en estimant la distribution jointe des variables d'entrées, ce qui peut nuire à la généralisation.

2.1 Réseaux de neurones

Une définition et description des réseaux de neurones artificiels à couches se retrouvent dans (Rumelhart et al., 1986c). Nous présentons seulement quelques points afin d'alléger la lecture du reste du chapitre.

Les modèles que l'on appelle réseaux de neurones artificiels s'inspirent de certaines observations faites sur le cerveau des animaux. Dans ces derniers, on observe une forte interconnectivité entre des cellules, les *neurones*, faisant individuellement un travail relativement simple. L'architecture et l'efficacité des connexions semblent expliquer le comportement de ce réseau de cellules. Un jeu subtil d'excitation et d'inhibition des signaux allant d'un neurone à l'autre conduit à l'exécution de tâches complexes chez l'animal. Dans les modèles artificiels, les connexions synaptiques sont représentées par des paramètres qui pondèrent l'intensité de la valeur émise par un neurone artificiel vers un autre auquel il est connecté. Comme chez le cerveau des animaux, le neurone artificiel joue le rôle d'intégrateur des différents signaux lui parvenant. Lorsque l'intensité du signal dépasse un certain seuil, ce neurone émet, à son tour, un signal qui sera propagé vers d'autres neurones. Si, dans le modèle

artificiel, la fonction est continue, comme c'est souvent le cas, le signal est émis avec une intensité variable.

Un réseau de neurones artificiel est composé d'un ensemble de fonctions simples (par exemple, la fonction sigmoïde ou la tangente hyperbolique), dites fonctions d'activation ou unités d'activation, reliées entres elles par un ensemble de connexions (dites poids ou paramètres) dont la valeur peut être modifiée suivant un algorithme, tel celui de la rétropropagation de l'erreur(Rumelhart et al., 1986b; LeCun, 1986). Lorsque le patron de connectivité permet de définir des sous-ensembles de neurones dont les membres ne partagent pas de connexions, nous parlons alors de réseaux à couches, ou à propagation avant ("feedforward"). La première couche est appelée *couche d'entrée*, la dernière est la *couche de sortie*, les autres, si elles sont présentes sont les *couches cachées*. Lorsqu'il y a des connexions des couches supérieures vers les couches inférieures, c'est-à-dire des cycles dans le graphe de connexions, nous parlons de *réseaux récurrents*. Un *réseau à délais* est un réseau de neurones dont certaines connexions comportent un délai de longueur τ. Une unité ayant été activée au temps t et ayant produit une réponse y, activera une unité à laquelle elle est reliée par une connexion de poids w comportant un délai τ au temps $t + \tau$ par la valeur $w\,y$. Une unité peut être connectée à elle-même. Les réseaux de neurones à délais sont souvent utilisés pour la reconnaissance de séquences (Hertz et al., 1991b; Bengio, 1996a). Des couches cachées effectuent des convolutions et du sous-échantillonnage durant le traitement des signaux. On les appelle les *TDNN* (time-delay neural networks). Dans l'article qui va suivre, il est question d'un *reverse-TDNN* (Simard and LeCun, 1992). C'est un *TDNN* où la résolution temporelle augmente de l'entrée vers la sortie et où les opérations d'échantillonage sont remplacées par des opérations d'interpolation.

Afin de résoudre une tâche, (classifier des images, trouver une fonction, reconnaître une séquence d'observations, etc.) dont nous avons des exemples, une manière de faire consiste à présenter successivement les exemples de la tâche aux unités d'entrées du réseau de neurones, de propager les signaux dans le réseau et de comparer la sortie

obtenue avec la sortie désirée. Selon un critère différentiable, nous rétropropageons l'erreur faite à travers tout le réseau de neurones. La règle que nous utilisons est celle de la dérivée en chaîne de la fonction de coût par rapport à tous les paramètres du réseau. C'est l'apprentissage dit supervisé d'un réseau de neurones artificiel.

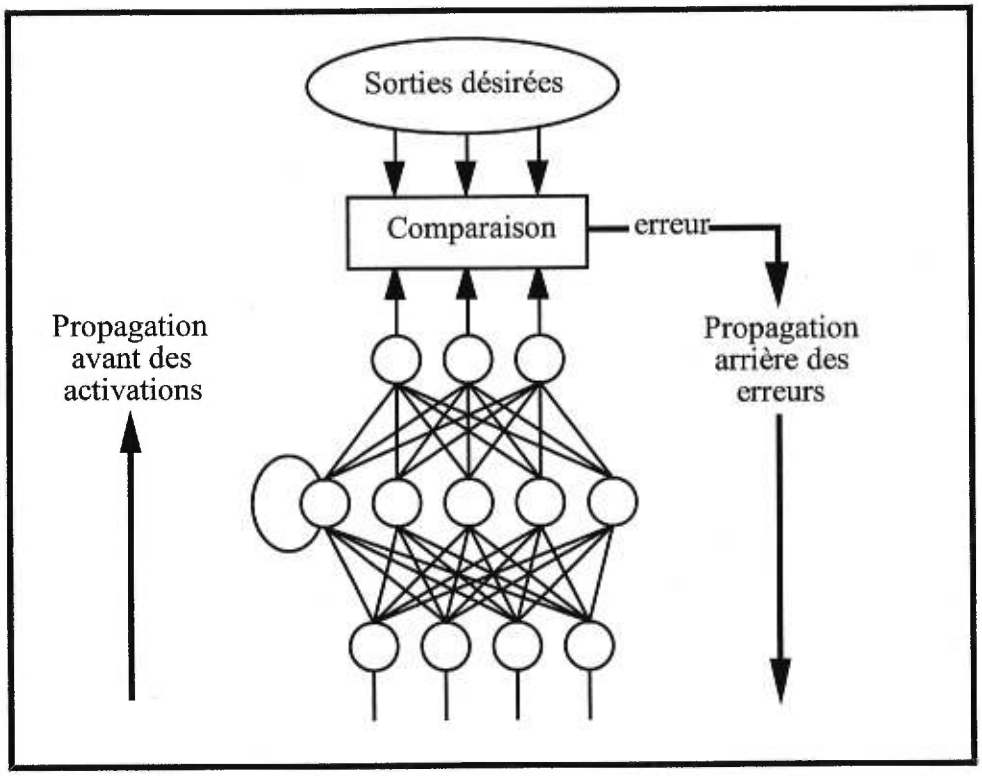

Figure 2.1. Exemple de réseau de neurones artificielles et illustration de la rétropropagation de l'erreur.

Cette méthode d'optimisation par descente du gradient ne garantit pas l'atteinte du minimum global de la fonction de coût. La valeur des poids vers laquelle convergera les poids du réseau de neurones peut correspondre à un minimun local de la fonction de coût. La solution trouvée peut donc dépendre de la valeur initiale des paramètres du réseau de neurones. Une façon de gérer ce problème consiste à répéter l'apprentissage plusieurs fois avec des valeurs de poids initiaux différentes. On peut aussi utiliser des

méthodes qui introduisent des perturbations dans la recherche de la solution, afin de ne pas demeurer dans un minimun local (Bottou, 1991; White, 1991).

La méthode que nous avons utilisée pour modifier les paramètres des réseaux de neurones récurrents est la rétropropagation à travers le temps (Back-Propagation Through Time). L'idée est de ramener un réseau récurrent arbitraire à un réseau à propagation avant. Cela consiste à déplier dans le temps le réseau récurrent autant de fois que la longueur de la séquence l'exige. Une fois déplié, on applique la méthode de rétropropagation du gradient (Rumelhart et al., 1986b; Hertz et al., 1991b; Nowlan, 1988)

Une partie des résultats contenus dans cet article furent présentés à la conférence *Neural Information Processing Systems 7* et publié dans les comptes rendus de la conférence NIPS-95. Il fut publié, dans la forme apparaissant ici dans *International Journal of Computational Intelligence and Organizations* en 1996.

Handling Asynchronous or Missing Data with Recurrent Networks

François Gingras **Yoshua Bengio**

gingras@iro.umontreal.ca **bengioy@iro.umontreal.ca**

514-343-6111x1794 **514-343-6804**

Dept. Physics **Dept. IRO**

Université de Montréal,

C.P. 6128, Succ. Centre-Ville, Montréal, Qc,

H3C-3J7, Canada

Acknowledgments

We would like to thank Sylvain Gauthier, Salah El HiHi and André Chabot for their collaboration, the NSERC and FCAR Canadian funding agencies as well as the IRIS institute for support.

Abbreviated title: **Missing Data with Recurrent Networks**

An important issue with many sequential data analysis problems, such as those encountered in financial data sets, is that different variables are known at different frequencies, at different times (asynchronicity), or are sometimes missing. To address this issue we propose to use recurrent networks with feedback into the input units, based on two fundamental ideas. The first motivation is that the "filled-in" value of the missing variable may not only depend in complicated ways on the value of this variable in the past of the sequence but also on the current and past values of other variables. The second motivation is that, for the purpose of making predictions or taking decisions, it is not always necessary to fill in the best possible value of the missing variables. In fact, it is sufficient to fill in a value which helps the system make better predictions or decisions. The advantages of this approach are demonstrated through experiments on several tasks.

2.2 Introduction

Learning from examples implies discovering certain relations between variables of interest. The most general form of learning essentially requires capturing the joint distribution of these variables. For example, in the case of classification problems, a traditional statistical approach is based on estimating the conditional distribution of the inputs for each class, together with the class prior probabilities (thus yielding the full joint distribution of inputs and classes). To minimize the number of classification errors, one picks the output class with the largest a posteriori probability given the input, assuming the joint model is correct. In cases where the training data are not abundant or non-stationary, the failure of that assumption can be harmful. It has been shown (Hampshire and Kumar, 1992) for classification tasks that this strategy is less optimal than one based on training the model with respect to the decision surfaces, which may be determined by a discriminant function assigned to each class (e.g., one output of a neural network for each class). The objective of training should be that the decision that is taken (e.g., picking the class whose corresponding discriminant function is the largest) has more chance of being correct, without necessarily assuming a particular probabilistic interpretation for the discriminant functions (model outputs). In that case, the model requires less parameters, as for example with a feedforward neural network trained to compute the expected output class probabilities given the observed variables. For a portfolio management application, we can consider the objective of learning to be the maximization of the return-risk ratio given the input variables (with specific constraints on the volume of transaction, the measure of risk, etc).

However, for many learning problems, only some of the input variables are given for each particular training case, and the missing variables differ from case to case. The simplest way to deal with this problem consists in replacing the missing values by their *unconditional mean* (Tresp et al., 1994; Ghahramani and Jordan, 1994). This

approach can be used with "discriminant" training algorithms such as those used with feedforward neural networks. However, in some problems, one can obtain better results by taking advantage of the dependencies among the input variables. A simple idea therefore consists in replacing the missing input variables by their *conditional* expected value, when the observed input variables are given. Many schemes have been proposed that essentially attempt to do this. Unfortunately, when the pattern of missing variables is arbitrary, this amounts to estimating the full joint distribution of all the variables. For example, with n_i inputs, capturing the possible effect of each observed variable on each missing variable normally requires $O(n_i^2)$ parameters (at least one parameter to capture some co-occurrence statistic on each pair of input variables). Many related approaches have been proposed to deal with missing inputs using a Gaussian (or Gaussian mixture) model (Ahmad and Tresp, 1993; Tresp et al., 1994; Ghahramani and Jordan, 1994). In the experiments presented here, a recurrent neural network is compared with a Gaussian mixture model trained with the expectation-maximization (EM) algorithm (Dempster et al., 1977) to handle missing values (Ghahramani and Jordan, 1994).

The present approach (discussed in section 2.3) is more economical than the traditional Gaussian-based approaches for two reasons. Firstly, we take advantage of hidden units in a recurrent network, which might be fewer than the inputs. The number of parameters depends on the product of the number of hidden units and the number of inputs. The hidden units only need to capture the dependencies among input variables which actually have some dependencies, *and* which are useful for reducing the output error. The second advantage is that training is indeed based on optimizing the desired criterion (e.g., reducing a classification error or maximizing the return), rather than predicting as well as possible the values of the missing inputs. The algorithm proposed in this paper does not depend on the pattern of missing variables (any subset of variables can be missing for any training pattern).

The recurrent network is allowed to relax for a few iterations (typically as few as

4 or 5) in order to fill-in some values for the missing inputs and produce an output. Relaxation describes the process by which a recurrent neural network, which is a dynamical system, is evolving toward a fixed point attractor (Hertz et al., 1991a). During the iterations of this relaxation, the observed inputs are kept fixed, while the other units of the network evolve, generally toward a fixed point. In section 2.4 we present experimental results with this approach, comparing the results with those obtained with a feedforward network and with a Gaussian mixture.

In section 2.5 we propose an extension of this scheme to sequential data. In this case, the network is not relaxing: inputs keep changing with time and the network maps an input sequence (with some missing values) to an output sequence. The main advantage of this extension is that it allows one to deal with sequential data in which the variables occur at different frequencies. This type of problem is frequent, for example with economic or financial data. Typically, the values of some variables are updated daily, others weekly, monthly or even yearly. Two experiments with such asynchronous data are presented in section 2.5. Finally, in section 2.6, we draw conclusive remarks.

2.3 A Relaxing Recurrent Network for Missing Inputs

Networks with feedback such as those proposed in (Pineda, 1987; Almeida, 1987; Pineda, 1989) and in the Boltzmann machine (Hinton et al., 1984) can be applied to learning a static input/output mapping when some of the inputs are missing. In both cases, however, one has to wait for the network to relax either to a fixed point (assuming it does find one) or to a "stable distribution" (in the case of the Boltzmann machine). In the case of fixed-point recurrent networks (Pineda, 1987; Almeida, 1987; Pineda, 1989), the training algorithm supposes that a fixed point has been reached. The gradient with respect to the weights is then computed in order to move the fixed point to a more desirable position. The approach we have preferred here avoids such an assumption. Instead, it uses a more explicit optimization of the whole behavior of the network as it unfolds in time, fills in the missing inputs and produces an output. The network is trained to minimize some function of its output, such as the mean squared error, by back-propagation through time (BPTT) (Rumelhart et al., 1986b). We have seen in our experiments that output units quickly converge, so we can afford to let the network relax for just a few iterations (5 in our experiments), thus speeding up both recognition and training. Because we use the BPTT algorithm, and do not make the assumption of actually reaching a fixed point, the algorithm works even if we do not wait until we have exactly reached a fixed point.

The algorithm works as follows. Suppose that a training pattern contains some missing components. The observed inputs are clamped whereas the missing inputs are initialized to their unconditional expectation but are then allowed to evolve using the feedback connections from the rest of the network. The input nodes associated with missing values for this particular pattern are treated like hidden nodes (see Figure 2.2).

During the relaxation, the network produces a sequence of outputs, one output for each step of the relaxation phase. To help the stability of the network and pre-

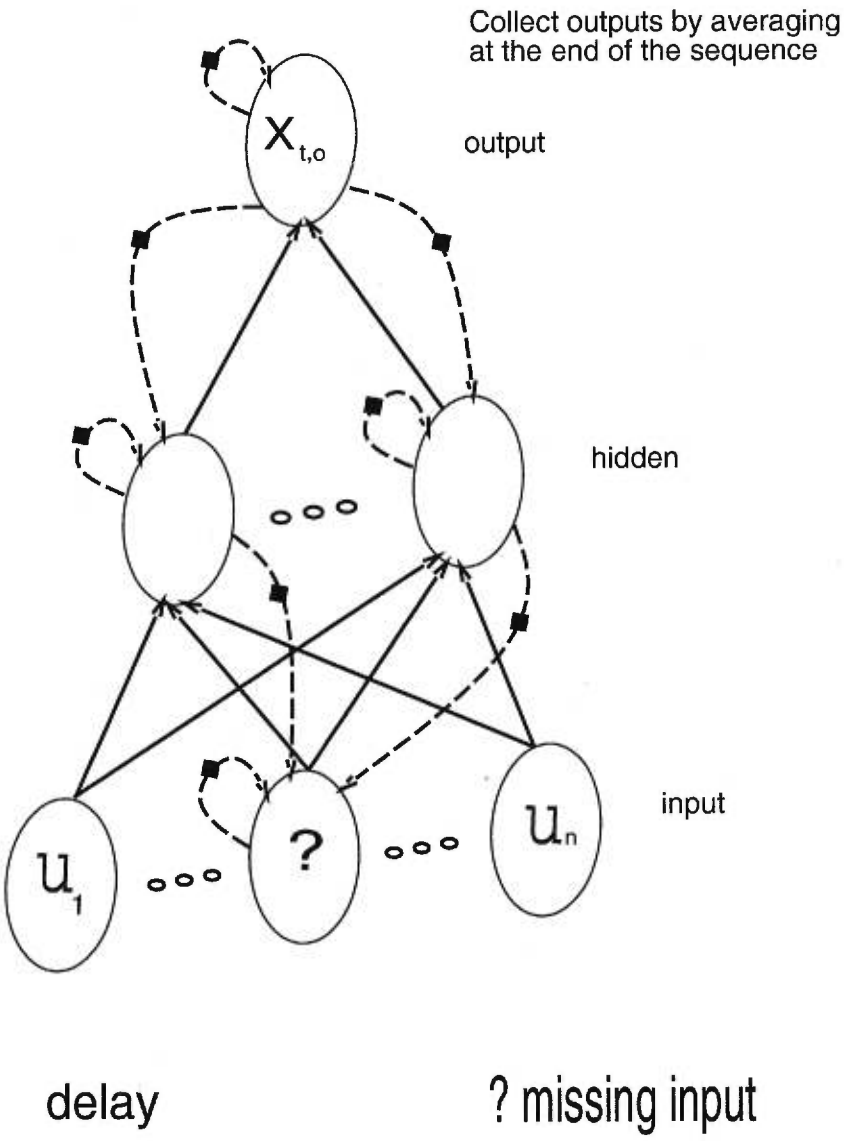

Figure 2.2.

Example of recurrent architecture for handling data with missing values. The recurrent links (dashed) toward the input layer are active only for the missing inputs.

vent it from finding periodic solutions (in which a correct output is generated only periodically), output supervision is given for every time step, with an error weighing scheme that puts more emphasis near the end of the sequence. Let us formalize this algorithm, which describes a module with an input vector u (which can contain missing values) and an output vector y.

Let U_I, U_H and U_O be the sets of input, hidden and output units respectively, and n_i, n_h, n_o denote the number of input, hidden, output units respectively, and n_u denotes the total number of units. u is the input vector, with u_i the i^{th} input value, which could take a special "missing" value. Let x_t be the vector $[x_{t,1}, \ldots, x_{t,n_u}]$ of activations of all the network units (including inputs, hidden and outputs), ordered such that if there is a zero-delay path in the network from unit i to unit j, then $i < j$. $I(i)$ and $O(i)$ are respectively the indices of the network units corresponding respectively to the i$^{\text{th}}$ element of the input or output vector. Let $E(i)$ denote the unconditional expectation of the i^{th} input variable.

In this paper hyperbolic tangent transfer functions are considered. A fixed vector v, with $v_t > 0$ and $\sum_t v_t = 1$ specifies a weighing scheme that distributes the responsibility for producing the correct output to the outputs at different time steps during the relaxation period. Its purpose is to encourage the network to develop stable dynamics which gradually converge toward the correct output (thus the weights v_t are chosen to gradually increase with t). The inertial term weighted by γ (in step 3 of the forward propagation algorithm below) is also used to help the network to find dynamically stable solutions. The parameter γ is fixed by hand. In the experiments described below, a value of 0.7 was used, but near values yielded similar results.

Please insert the algorithm box here

Let us define a **module** as a subsystem having inputs and outputs. If we can

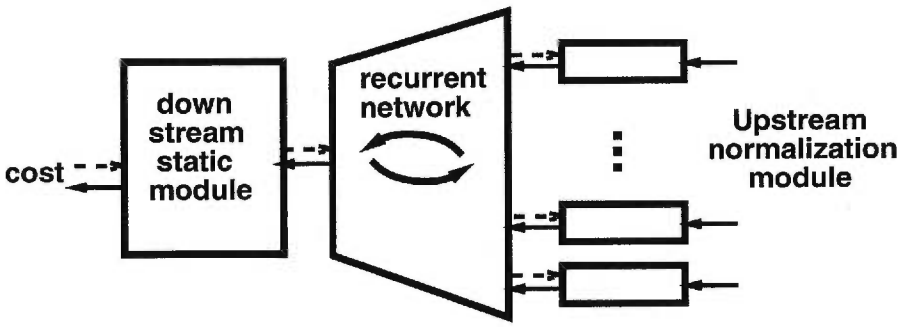

Figure 2.3.

Example of a hybrid modular system, using the recurrent network (middle) to extract features from patterns which may have missing values. It can be combined with upstream modules (e.g., a normalizing preprocessor, right) and downstream modules (e.g., a static classifier, left). Gradients with respect to a cost function can be propagated through all the modules. Dotted arrows show the backward flow of gradients.

compute the first derivative of the module's output with respect to each of its inputs and with respect to its parameters, we can train any combination of modules to optimize a global criterion. It can be shown that training such modular systems directly with respect to the desired global criterion (rather than training them separately) yields better optimization of this criterion (Bengio, 1996a). The derivative of this criterion with respect to the parameters of the module can be used for improving the behavior of this module. The gradient with respect to the inputs of a module are used to propagate error information to other modules whose output influences the input of this module (Bottou and Gallinari, 1991).

For example, as in Figure 2.3 there might be another module taking as input the recurrent network's output.

In this case the recurrent network can be seen as a feature extractor that accepts data with missing values in input and computes a set of features that are never missing. In another example of hybrid system the non-missing values in input of

the recurrent network are computed by another upstream module (such as the pre-processing normalization used in our experiments), and the recurrent network would provide gradients to this upstream module (for example, to better tune its normalization parameters). See (Bottou and Gallinari, 1991) for more examples of building a modular system trained with a gradient-based learning algorithm.

2.4 Experiments with Static Data

2.4.1 A Classification Task

A network with three layers (input, hidden, output) was trained to classify data with missing values from the *audiology* database. This database was made public thanks to Jergen and Quinlan, was used by (Bareiss and Porter, 1987), and was obtained from the UCI Repository of machine learning databases (`ftp.ics.uci.edu:pub/machine-learning-databases`).

The original database has 226 patterns, with 69 attributes, and 24 classes. Unfortunately, most of the classes have only 1 sample. Hence we have decided to group the classes into four clusters. To do so, the average pattern for each of the 24 classes was computed, and the K-Means algorithm (MacQueen, 1967) was then applied on these 24 class prototypes to yield the 4 clusters used in our experiments. The multi-valued input symbolic attributes (with more than 2 possible values) were coded with a "one-out-of-n" scheme, using n inputs (all zeros except the one corresponding to the attribute value). Note that a missing value is represented with a special numeric value recognized by the neural network module. The input features which were constant over the training set were removed. The remaining 90 inputs were standardized (by computing mean and standard deviation) and transformed by a saturating non-linearity (a scaled hyperbolic tangent) (LeCun et al., 1991). Standardization makes the average small with respect to the variance, which helps to speed up convergence of the network, by improving the conditioning of the Hessian. The Hessian matrix is the matrix of second derivative of the optimization criterion with respect to the parameters, its condition number being the ratio between the magnitudes of its largest and smallest eigenvalues. The convergence time can be shown to be proportional to that number (Becker and LeCun, 1989). For a neural network, we must avoid too large absolute values of inputs, which would stall the gradient descent-based learning algorithms, because the derivatives through the saturated non-linearities of the acti-

vation functions would be close to zero. Since the distributions of the variables are not exactly normal, some large values remain. Therefore, the above standardization step was followed by a soft clipping step: the standardized features were squashed with a scaled hyperbolic tangent, so that values beyond 3 standard deviations would essentially be saturated. For the same reason, we prefer to use a hyperbolic tangent activation function rather than the symetric sigmoid (logistic). The output class is coded with a "one-out-of-4" scheme, and the recognized class is the one for which the corresponding output has the largest value.

The architecture of the network is depicted in Figure 2.4. The length of each relaxing sequence in the experiments was 5. Higher values would not bring any measurable improvements, whereas for shorter sequences performance would degrade. The number of hidden units was varied, with the best generalization performance obtained using 3 hidden units.

The recurrent network was compared with feedforward networks as well as with a mixture of Gaussians. For the feedforward networks, the missing input values were replaced by their unconditional expected values. The feedforward neural networks were trained to minimize the same criterion as the recurrent networks, i.e., the sum of squared differences between network output and desired output. Several feedforward neural networks with varying numbers of hidden units were trained. The best generalization (lowest classification error) was obtained with 15 hidden units. Experiments were also performed with no hidden units and two hidden layers (see Table 2.1). Note that this series of experiments does not constitute a random or exhaustive search in the space of architectures, but instead follows the principles of structural risk minimization (Guyon et al., 1993), taking advantage of the relation between capacity and generalization.

The recurrent network was also compared with an approach based on a Gaussian and Gaussian mixture model of the data. We used the algorithm described in (Ghahramani and Jordan, 1994) for supervised learning from incomplete data

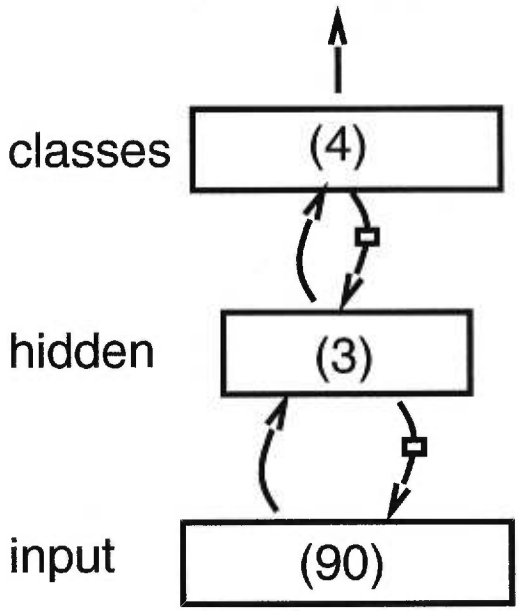

Figure 2.4.

Architecture of the recurrent networks in the static data experiments. Small squares represent a unit delay. The number of units in each layer is reported inside the rectangles.

with the EM algorithm. The whole joint input/output distribution is modeled using a mixture model with Gaussians (for the inputs) and multinomial (outputs) components:

$$P(\mathbf{X} = \mathbf{x}, \text{Class} = c) = \sum_j P(\omega_j) \frac{\mu_{jd}}{(2\pi)^{n/2}|\Sigma_j|^{1/2}} \exp\{-\frac{1}{2}(\mathbf{x} - \mu_j)'\Sigma_j^{-1}(\mathbf{x} - \mu_j)\}$$

where $\mathbf{x}$ is the input vector, c the output class, and $P(\omega_j)$ the prior probability of component j of the mixture. The μ_{jd} are the multinomial parameters; μ_j and Σ_j are the Gaussian mean and covariance for component j. Maximum likelihood training is applied as explained in (Ghahramani and Jordan, 1994), taking missing values into account (as additional missing variables of the EM algorithm).

For each architecture in Table 2.1, ten training trials were run with a different subset of 200 training and 26 test patterns (which is the original train/test division proposed in the UCI repository) and different initial weights for the neural networks. Although there are more parameters than data points, we used early stopping to prevent overfitting: the effective capacity of the network (taking into account the smaller number of training iterations) is therefore much less than the maximum capacity. The early stopping procedure has already proved to be a useful way to avoid overfitting (Chauvin, 1990; Morgan and Bourlard, 1990).

The recurrent network was superior to the other architectures, probably for the reasons discussed in the conclusion. In addition, we have graphically shown the rate of convergence during training of the best feedforward network (90-15-4) as well as the best recurrent network (90-3-4), in Figure 2.5. Clearly, the recurrent network not only performs better at the end of training but also learns much faster.

2.4.2 Portfolio Management

In this section, we compare the recurrent and static networks on the following measure of performance: the return of a portfolio minus the return of a reference strategy. Portfolio management consists in selecting a sequence of portfolio weights in order

Table 2.1.

Comparative performance of recurrent networks, feedforward networks, and Gaussian mixtures density models on audiology data set. The average percentage of classification error is shown for both training and test sets, together with the standard deviation in parenthesis, for 10 trials.

	Training set error	Test set error
90-3-4 Recurrent net	0.3(0.6)	2.7(2.6)
90-6-4 Recurrent net	0(0)	3.8(4)
90-25-4 Feedforward net	0.5(1.6)	15(7.3)
90-15-4 Feedforward net	0.8(0.4)	13.8(7)
90-10-6-4 Feedforward net	1(0.9)	16(5.3)
90-6-4 Feedforward net	6(4.9)	29(8.9)
90-2-4 Feedforward net	18.5(1)	27(10)
90-4 Feedforward net	22(1)	33(8)
1 Gaussian	35(1.6)	38(9.3)
4 Gaussians Mixture	36(1.5)	38(9.2)
8 Gaussians Mixture	36(2.1)	38(9.3)

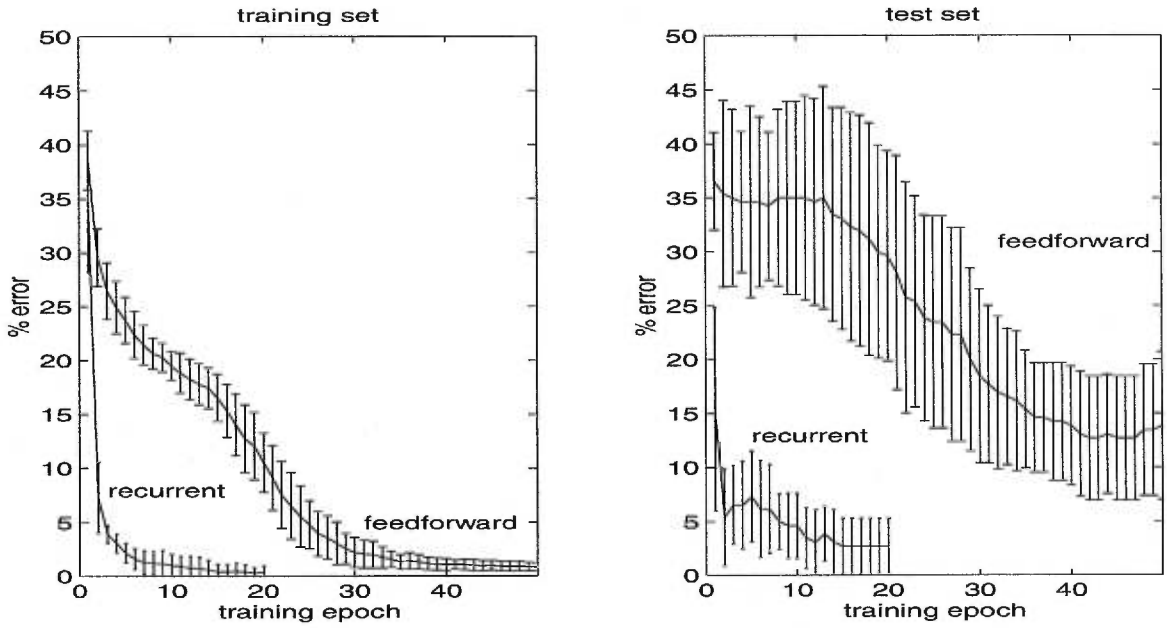

Figure 2.5.

Evolution of training and test error for the recurrent network and for the best of the feedforward networks (90-15-4): average classification error with respect to training epoch, (with 1 standard deviation error bars, computed over 10 trials).

to optimize some financial criterion. At each time step (here every month), we must decide how to distribute our worth among assets in order to maximize profit. We built a monthly database of 2 macroeconomic indicators and 3 stock-specific variables, for the period Dec. 84 - Feb. 95. The features represent macro-economic variables which are known to influence the business cycle, and micro-economic variables representing the value and profitability of the company. The micro-economic variables are only given every 12 months because they depend on yearly company statements. Because of the non-stationarity and temporal dependency of the data, we can obtain better results with the static network if we fill in the missing data using a cubic spline extrapolation rather than with their unconditional expectation. Each variable was then made uniform (by taking its rank in a two year window). The portfolio has 36 assets: 35 Canadians stocks and 3-month treasury bills. The 35 companies are major companies of the Toronto Stock Exchange (most of them in the TSE35 Index). The decisions to buy or sell assets were taken in order to minimize transactions, maximize diversification, and maximize the ratio of return to risk, as explained in (Bengio, 1996b).

A 5-3-1 network was used for all 35 stocks, with a single output representing expected future returns over a 3-month horizon. Preliminary experiments with the network architecture suggested that using approximately 3 hiddens units yielded better results than using no hidden layer or many more hidden units. Better results might be obtained by considering different sectors of the market (different types of companies) separately, but the purpose of the experiments reported here was mainly to compare algorithms in the presence of missing data. The parameters of the network were therefore shared across time and across the 35 stocks. The 36^{th} output (for expected return of cash) was obtained from the current short-term interest rates. We used the data of the first 72 months for training, keeping the remaining 48 months for the test set. The curve of the relative return, shown in Figure 2.6, is the mean over ten trials with different initial weights.

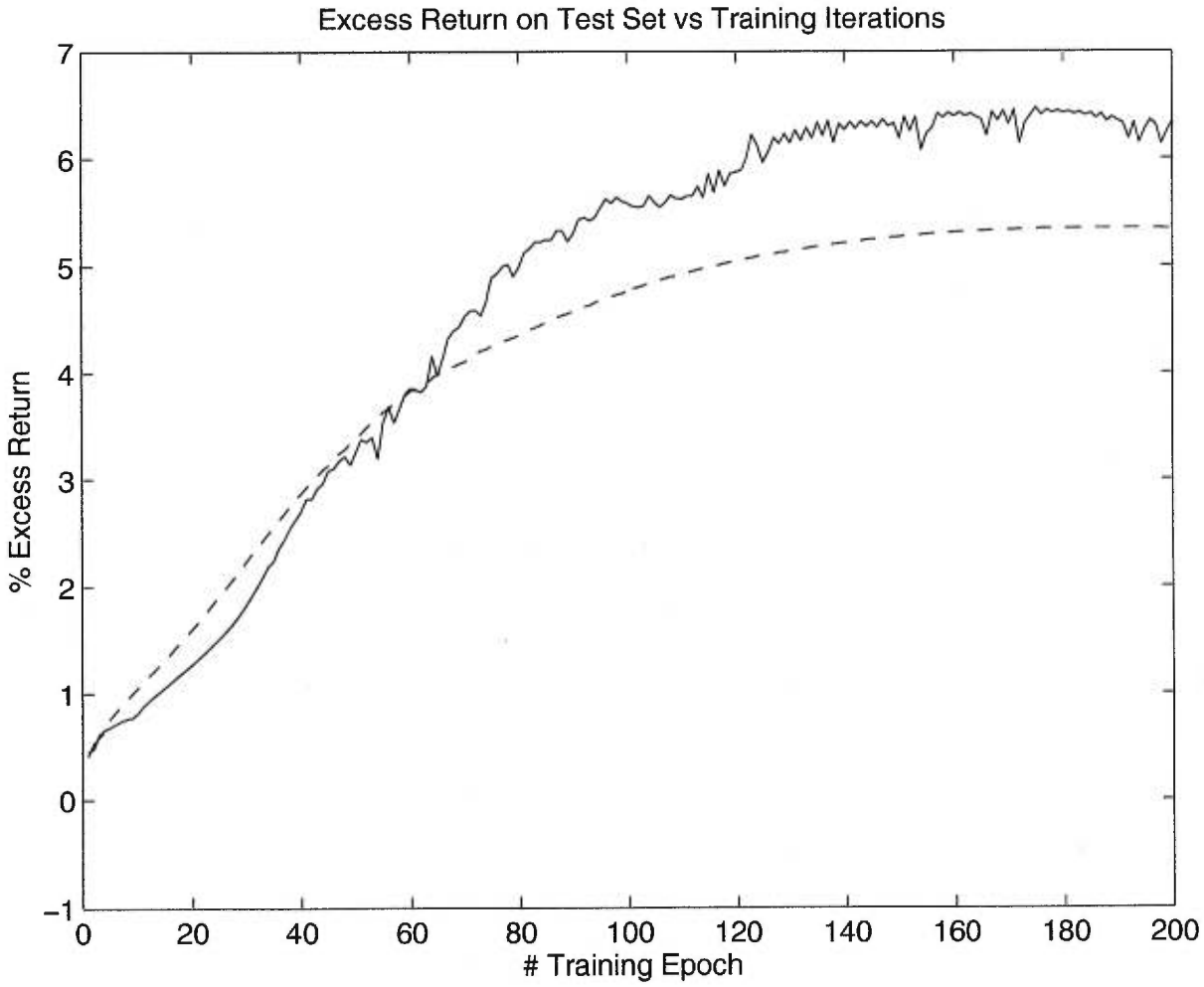

Figure 2.6.

Evolution of the mean excess return on porfolio management over the test data. Dashed: static network. Hard: recurrent network with feedback to input values to complete missing data.

Instead of using the prediction criterion (mean-squared error), we trained the network jointly with a trading module, in order to maximize profit, taking transaction costs into account (Bengio, 1996b). This procedure yielded about 4% better yearly return on the same data than the mean squared error criterion. Training was performed for 200 epochs. A buy-and-hold benchmark was used to compare the networks with a conservative policy. For this benchmark, the initial portfolio is distributed equally among all the stocks (with no cash). Then there are no transactions. The return of the benchmark is measured in the same way as that of the network (Bengio, 1996b), but with no transactions. The excess return is the difference between the overall return obtained by a network and that of the buy-and-hold benchmark.

For the experiments with the recurrent network, we used a relaxing sequence of length 5. At the first time step of the relaxation, the input was initialized to the same value used in the case of the static network, while during the remaining time steps the network updated the missing inputs.

Results are summarized in Figure 2.6, showing the evolution of excess return in the test period, during training, for both the static and recurrent networks, averaged over 10 trials. On the test set, the return of the benchmark was 6.6%. The recurrent network yielded 12.9% return on average, i.e., 6.3% excess return. The static network yielded 11.9% return on average, i.e., 5.3% excess return. The use of a recurrent network to fill in the missing values improves excess return by about 1% which can be considered interesting in light of the present task.

2.5 A Recurrent Network for Asynchronous Sequential Data

An important consideration with many sequential data analysis problems such as those encountered in financial applications is that different variables are known at different frequencies, at different times (phase), or are sometimes missing. For example, some variables are given daily, weekly, monthly, quarterly, or yearly. Furthermore, some variables may not even be given for some of the periods or the precise timing may change (for example, the date at which a company reports financial performance may vary).

Therefore, we propose to extend the algorithm presented above for static data with missing values to the general case of sequential data with missing values or asynchronous variables. For time steps at which a low-frequency variable is not given, the input is considered as a missing value. In the framework proposed here, asynchronous time-series are viewed merely as time-series with the same missing values (for the low frequency variables). For example, if we are considering two time series, one with monthly data and the other with quarterly data, we consider the latter like a monthly time series with a missing value two months out of three. The feedback links from the hidden and output units to the input units allow the network to "complete" the missing data. The main differences with the static case are that the inputs and outputs vary with t (we use u_t and y_t at each time step instead of u and y). The training algorithm is otherwise the same.

To evaluate the algorithm, we have made two experiments, one with artificially generated data (a more controlled situation), and the other, with financial data containing both quarterly time series and monthly time series.

2.5.1 Experiments with Data Generated by a Recurrent Network

The objective of this experiment is to compare the performance of the proposed algorithm to that of a static (time delay) neural network, on a dataset which can be

controlled and for which a large number of training and test sets can be generated. For this experiment, data was generated by a recurrent network with random weights and feedback links on the input units. The generating network has 6 inputs, 3 hidden units and 1 output. The hidden layer is connected to the input layer with one time delay. The hidden layer receives inputs with delays 0 and 1 from the input layer and with delay 1 from itself. The output layer receives inputs from the hidden layer. At the initial time step and at 5% of the time steps (chosen randomly), the input units were clamped with uniform random values around 0 to introduce some further variability. To generate asynchronous data, half of the inputs were also hidden (i.e. made missing) 4 out of every 5 time steps. 100 training sequences and 50 test sequences were generated. The learning problem is therefore a sequence regression problem with missing and asynchronous input variables. The network used to generate the data is different from the network used to learn the data.

Comparative experiments show a clear advantage to completing the missing values (due to the the different frequencies of the input variables) with a recurrent network, as shown in Figure 2.7.

The recognition recurrent network, shown in Figure 2.8, has a different architecture from the generating network. It has multiple time scales (implemented with subsampling and oversampling, as in TDNNs (Lang and Hinton, 1988; Waibel et al., 1989) and reverse-TDNNs (Simard and LeCun, 1992)), to facilitate the learning of such asynchronous data. The static network is a time-delay neural network with 6 input units, 8 hidden units, and 1 output unit, and connections with delays 0, 2, and 4 from the input to hidden and hidden to output units. The "missing values" for slow-varying variables were replaced by the last observed value in the sequence. Experiments with 4 and 16 hidden units yielded similar results. The experiment confirms that when there are significant dependencies among the observed and missing variables (considering also their past values), some of these dependencies can be captured by the recurrent network, and this yields better performance than with the

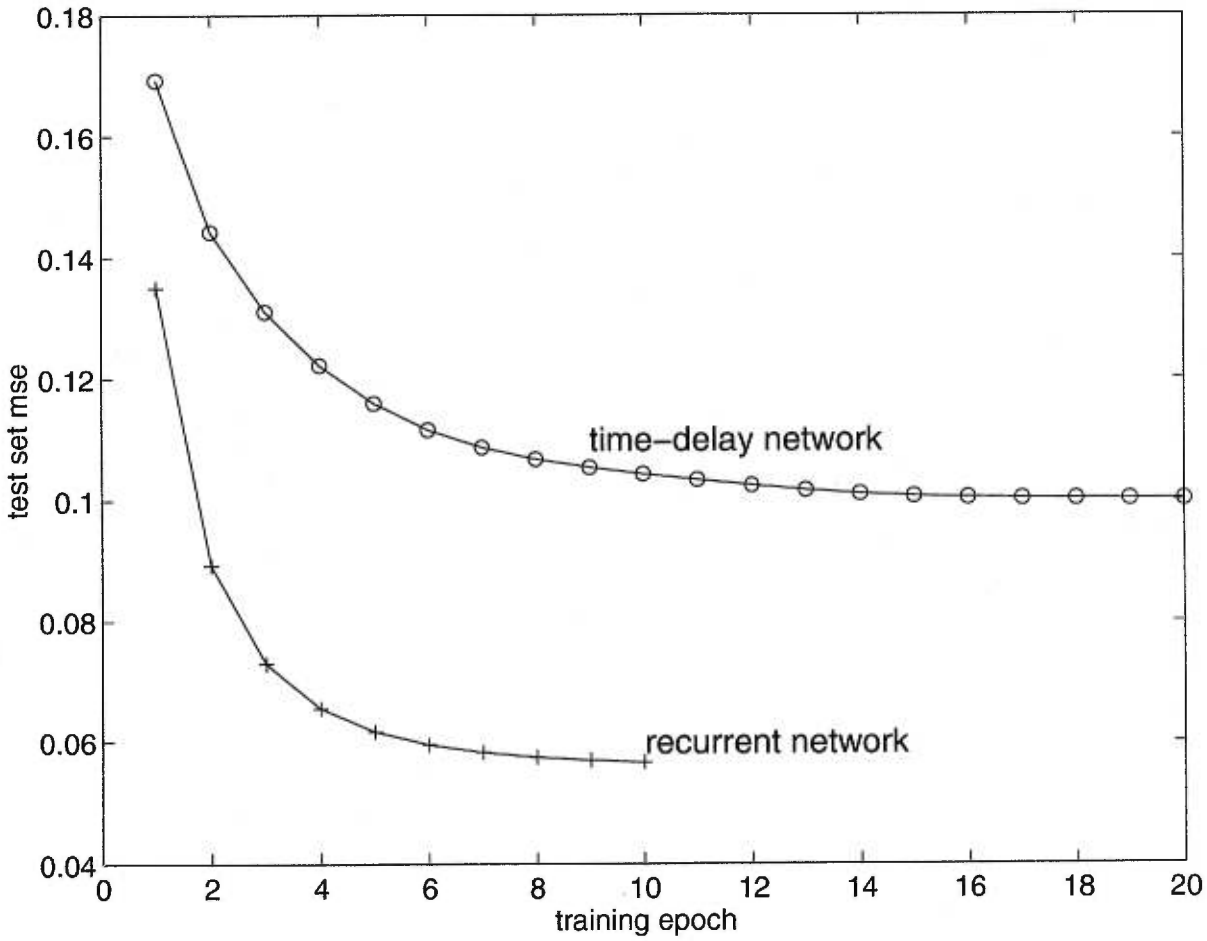

Figure 2.7.

Test set mean squared error on the asynchronous data. Top: static network with time delays. Bottom: recurrent network with feedback to input values to complete missing data.

static network.

2.5.2 Financial Data

For this experiment, we used monthly and quarterly economic time series in order to predict the 12-month change in the TSE300, the main index of the Toronto Stock Exchange (an aggregate of the prices of 300 stocks). For this experiment, the inputs were chosen from a set of economic time series, ranging from 1959 to 1993. For the training set we used the data from January 1959 to January 1986, and for the test set, the data from February 1986 to December 1993.

We started from a set of 34 time series representing the main economic indicators believed to influence the Canadian stock market. Out of these series, we generated a larger set by computing rates of change over 12 months as well as moving averages of these rates of change over 3 and 6 months. Using experiments with static feedforward neural networks, we ranked these series and selected a subset. The algorithm we used for selecting them is the following. We first selected the single most predictive series of the set (by training a linear predictor with one input, for each of these series). Then we selected a second series by training another set of networks (with 1 hidden unit) for each possible second input series. We repeated this process with the 3rd, 4th, etc..., increasing the number of hidden units to a maximum of 4 along the way. In total, $N(N-1)/2$ training experiments are therefore performed, given N series to start with. To compare the advantage of using one series instead of another, one does not need to wait for a full training; the first few epochs are generally quite indicative. Therefore those training experiments were all stopped after a fixed number of epochs (8). With each subset of series we also obtained a generalization score, which tended to worsen for larger sets of series. In the end, we selected a set of 34 series for the experiments to follow (with recurrent networks). 10 of these series were quarterly and the remaining were monthly. For the quarterly series, the input to the recurrent network was left free (special "missing value") except for those months (one out of 3)

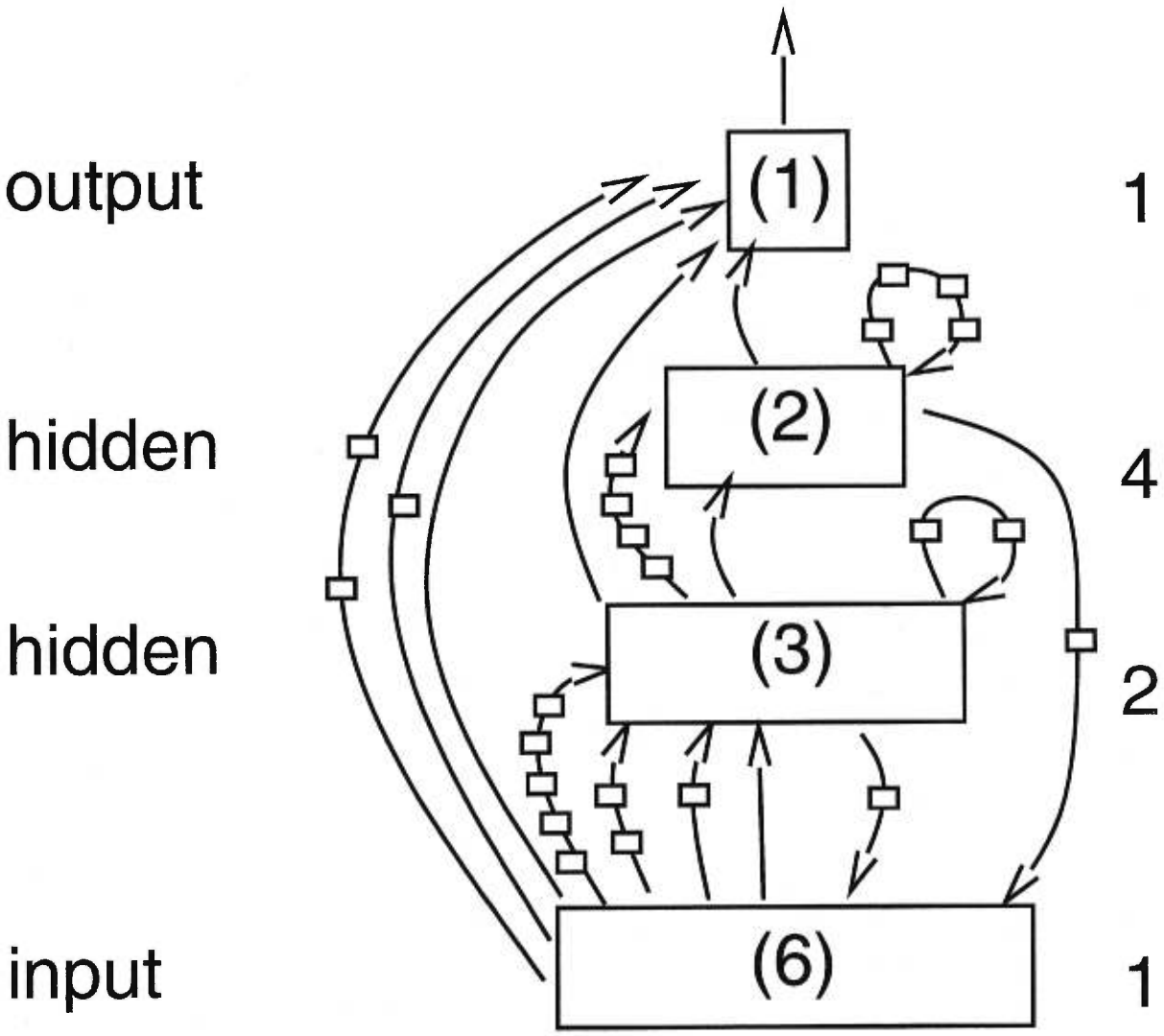

Figure 2.8.

A 6-3-2-1 architecture with multiple time scales for asynchronous sequential data. Small squares represent a unit delay. The number of units in each layer is inside the rectacles. The time scale at which each layer operates is on the right of each rectangle.

when its corresponding economic variable was made known to the market (**not** when the corresponding economic variable actually attained this value, which would be several months before its announcement). Note that not all those quarterly variables had the same phase.

As before, we performed some further preprocessing in order to standardize the inputs and prevent outliers from disrupting network training. Soft clipping prevents outliers (with large values) from saturating hidden units. For this purpose, and to take into account the non-stationarities in the average of these series, a weighted moving average linear filter was computed for each series. The weights (adding to 1) linearly increase in time to put more emphasis on recent history. These moving averages were then substracted from each series. A moving standard deviation was similarly computed and used to normalize the amplitude of the series.

In preliminary experiments we tried to predict the exact monthly change in the TSE, minimizing the squared difference between predicted and actual change. Results were not very good, in part because the desired output is a very noisy signal. We found that more reliable results could be obtained by asking less precision from the neural network, i.e., asking it only to approximate the desired change, using a discretization of the output variable into five categories. The prediction problem was therefore turned into one of classification. The ranges of values for each category were decided by inspection of the distribution of the output variable (to obtain approximately the same number of observations in each category). The five categories are the following:

1. less than -9%: strong loss,

2. between -9% and -3%: moderate loss,

3. between -3% and 3%: no significant change,

4. between 3% and 9%: moderate gain,

5. above 9%: strong gain.

The networks therefore had 5 output units for those 5 classes.

Because of the non-stationarity in the data, all the networks were trained with a weighted mean squared error criterion, putting more emphasis on recent history:

$$C = \sum_t w_t(\hat{y}_t - y_t)^2$$

where the w_t are fixed positive weights linearly increasing with t, $\hat{y}_t$ is the network output vector for time step t, and y_t is the corresponding desired output.

A recurrent network similar to the one shown on Figure 2.4 was used, but with 2 hidden units and 5 outputs. For the experiment with the static networks, the missing values were replaced by their previously known value.

Because of the scarcity of data, both the recurrent network and the static network tend to overfit if training lasts too long or if the number of hidden units chosen is larger than 2 or 3. In both cases the best results were obtained with 2 hidden units. The (5-way) classification error on the test set went down to 31.2% for the static network and to 26.8% for the recurrent network.

2.6 Conclusion

When there are dependencies among input variables, and the output prediction can be improved by taking them into account, we have seen that a recurrent network with input feedback can perform significantly better than a simpler approach that replaces missing values by their unconditional expectation or simple extrapolation. According to us, this explains the significant improvement brought about by using the recurrent network instead of a feedforward network in the experiments.

On the other hand, the large number of input variables ($n_i = 90$, in some of the experiments) most likely explains the poor performance of the mixtures of Gaussian models in comparison to both the static networks and the recurrent networks. The Gaussian models require the estimation of $O(n_i^2)$ parameters and inverting large covariance matrices.

The approach to handling missing values presented here has also been extended to sequential data with missing or asynchronous variables, which is a frequent problem with financial data sets. As our experiments suggest, for such problems, using recurrence and feedback into the inputs yields better performance than static networks for which the missing values are filled using a heuristic.

We believe these preliminary experiments open the way to a large number of applications both with static and sequential data in which the variables are missing or given at different time scales.

Computation of Outputs Given Observed Inputs

Given: input vector $u = [u_1, u_2, \ldots, u_{n_i}]$,

E(i) = unconditional expectation for i$^{\text{th}}$ input

Result: output vector $\hat{y} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{n_o}]$

1. Initialize for $t = 0$:

   For $i = 1 \ldots n_u$, $x_{0,i} \leftarrow 0$

   For $i = 1 \ldots n_i$, if u_i is missing then $x_{0,I(i)} \leftarrow E(i)$,

   Else $x_{0,I(i)} \leftarrow u_i$.

2. Loop over time:

   For $t = 1 \ldots T$

   For $i = 1 \ldots n_u$

   If $i = I(k)$ is an input unit and u_k is not missing then

   $x_{t,i} \leftarrow u_k$

   Else

   $x_{t,i} \leftarrow (1 - \gamma) x_{t-1,i} + \gamma f(\sum_{l \in S_i} w_l x_{t-d_l,p_l})$

   where S_i is a set of links from unit p_l to unit i,

   each with weight w_l and a discrete delay d_l

   (terms for which $t - d_l < 0$ are not considered).

3. Collect outputs by averaging at the end of the sequence:

   $\hat{y}_i \leftarrow \sum_{t=1}^{T} v_t \, x_{t,O(i)}$

Back-Propagation

The back-propagation computation requires an extra set of variables $\dot{x}_t$ and $\dot{w}$, which will contain respectively $\frac{\partial C}{\partial x_t}$ and $\frac{\partial C}{\partial w}$ after this computation, where C is the cost function.

`Given:` `output gradient vector` $\frac{\partial C}{\partial \hat{y}}$

`Result:` `input gradient` $\frac{\partial C}{\partial u}$ `and parameter gradient` $\frac{\partial C}{\partial w}$.

1. `Initialize unit gradients using outside gradient:`

   Initialize $\dot{x}_{t,i} = 0$ for all t and i.

   For $i = 1 \ldots n_o$, initialize $\dot{x}_{t,O(i)} \leftarrow v_t \frac{\partial C}{\partial \hat{y}_i}$

2. `Backward loop over time:`

   For $t = T$ to 1

   For $i = n_u \ldots 1$

   If $i = I(k)$ is an input unit and u_k is not missing then

   no backward propagation

   Else

   For $l \in S_i$

   If $t - d_l > 0$

   $$\dot{x}_{t-d_l,p_l} \leftarrow \dot{x}_{t-d_l,p_l} + (1-\gamma)\dot{x}_{t-d_l+1,p_l} + \gamma w_l \dot{x}_{t,i} f'(\textstyle\sum_{\lambda \in S_i} w_\lambda x_{t-d_\lambda,p_\lambda})$$

   $$\dot{w}_l \leftarrow \dot{w}_l + \gamma f'(\textstyle\sum_{\lambda \in S_i} w_\lambda x_{t-d_\lambda,p_\lambda}) x_{t-d_l,p_l}$$

3. `Collect input gradients:`

   For $i = 1 \ldots n_i$,

   If u_i is missing, then $\frac{\partial C}{\partial u_i} \leftarrow 0$

   Else $\frac{\partial C}{\partial u_i} \leftarrow \sum_t \dot{x}_{t,I(i)}$

Chapitre 3

SECOND ARTICLE: CLASSIFICATION DE SIGNAUX AVEC UNE MIXTURE CONTRAINTE

Dans ce chapitre, nous abordons un autre problème qui peut survenir dans la modélisation en général et la modélisation de données séquentielles en particulier. Le problème auquel nous sommes confrontés, que nous exposerons en détail plus bas, est complexe. Nous avons très peu d'exemples différents de la relation entre les données et les classes des exemples ne sont pas clairement identifiés. De plus le modèle à construire doit nous permettre d'identifier des données qui correspondent à des cas atypiques, c'est-à-dire des cas qui ne correspondent pas à des situations déjà rencontrées, et donc pas dans l'ensemble d'apprentissage.

3.1 Description du problème

Dans un réacteur nucléaire de type CANDU (Canadian Deuterium), divers senseurs sont répartis afin de mesurer différentes variables utilisées pour suivre l'évolution des propriétés thermo-hydrauliques et des caractéristiques de puissance du réacteur. Parmi ces variables, on retrouve la température, la pression, le flux du liquide de refroidissement. Les données provenant des senseurs sont supposées représenter quatre états, ou régimes du réacteur. Ces données sont collectées dans des fichiers, et nous savons que dans un fichier en particulier, des données associées à un certain régime sont présentes. De plus, nous savons que la notion de régime n'est pas toujours clairement définie: à un régime donné, il peut correspondre une gamme de valeurs possibles pour certaines variables. Aussi, dans un fichier, plus d'un régime peut être présent, ceci étant dû à la séquence des opérations entourant les manoeuvres des opérateurs

responsables du réacteur.

Les dizaines de milliers de points contenus dans les fichiers de données ne constituent pas, à vrai dire, un grand ensemble d'exemples pour l'apprentissage. En effet, la plupart de ces points sont une réalisation bruitée du même signal, car le réacteur passe plusieurs heures dans le même état avant d'effectuer une transition vers un autre état. De plus, la présence de divers types de senseurs, chacun mesurant une variable particulière, ne semble pas fournir autant de dimensions indépendantes pour l'analyse de l'état du réacteur. Il existe une certaine corrélation entre les différentes variables observées, corrélation qui peut varier selon l'état du réacteur comme le laisse voir l'inspection des différents signaux.

Nous devons donc proposer un modèle à partir d'une base de données qui contient très peu d'exemples de chacun des états du réacteur. Nous pouvons difficilement, dans ce cas, explorer un vaste espace de solutions possibles, car la composante variance du terme d'erreur de généralisation serait très grande. Nous imposons donc des contraintes sur la forme de la solution possible (Baxter, 1996). Ces contraintes nous sont suggérées par la connaissance que nous avons du contenu des fichiers.

Nous avons décidé de construire un classifieur d'état, c'est-à-dire un modèle qui établit une correspondance entre un vecteur d'observations et une étiquette ou classe. Le problème de la classification est bien exposé dans(Duda and Hart, 1973). Une des classes que le modèle doit recommander lors de son utilisation est la classe *rejet.* En examinant la valeur relative de la vraisemblance des données, nous pouvons nous donner un certain confort avec la décision rendue par le modèle. Cela diffère de la manière habituelle de faire du rejet en regardant la valeur de la probabilité a posteriori (étant donné un vecteur d'observations et un modèle). Permettre le rejet améliore les performances de classification, mais nous devons prendre soin de ne pas rejeter indûment des cas typiques et importants d'identifier.

Cet article fut publié dans *Computers and Artificial Intelligence,* **17**(2-3):189-210,1998.

Gaussian Mixture Densities for Classification of Nuclear Power Plant Data

Y. Bengio[1], F. Gingras[1,2], B. Goulard[2], J.-M. Lina[2,3] and K. Scott[3]

1: LISA, Dept. d'Informatique et de Rech. Operationnelle, Univ. de Montréal
C.P. 6128 Succ. Centre-Ville, Montréal(Québec), H3C 3J7, Canada

2: PHYSNUM, Centre de Recherches Mathematiques, Univ. de Montréal
C.P. 6128 Succ. Centre-Ville, Montréal(Québec), H3C 3J7, Canada

and

3: Atlantic Nuclear Services Ltd., Fredericton, New Brunswick, E3B 5C8, Canada

Abstract

In this paper we are concerned with the application of learning algorithms to the classification of reactor states in nuclear plants. Two aspects must be considered: (1) some types of events (e.g., abnormal or rare) will not appear in the data set, but the system should be able to detect them, (2) not only classification of signals but also their interpretation are important for nuclear plant monitoring. We address both issues with a mixture of mixtures of Gaussians in which some parameters are shared to reflect the similar signals observed in different states of the reactor. An EM algorithm for these shared Gaussian mixtures is presented. Experimental results on nuclear plant data demonstrate the advantages of the proposed approach with respect to the above two points.

3.2 Introduction

One of the basic problems in the validation of data from a nuclear power plant is to estimate the state of the reactor from noise-contaminated data. In a previous work (Dai et al., 1995), a wavelet-based pre-processing of the channel outlet temperature signal has been used for detection of flow blockage during the start-up operation of the PLGS nuclear plant. So far, this work was concerned with the detection of characteristic transients. In this paper, we address the problem of *classification of reactor states*. The data are time sequences collected from various sensors. For sake of illustration of the technique presented here, we choose 3 observed signals and 4 main classes: normal state, refuelling operation, low power regime and governor valve action. An extra "unknown class" represents unseen (e.g. rare or abnormal) types of signals (when the probability of the data given the model is very low), and we will use this for testing the model proposed here. The ability to detect such abnormal states despite the absence of training examples of such states is clearly very important for nuclear plant monitoring.

Previous work in the application of learning algorithms (such as artificial neural networks) to nuclear plants concentrate on monitoring the state of the plant (Bartlett and Uhrig, 1992; Basu and Bartlett, 1994; Parlos et al., 1994; Kozma et al., 1996), identifying transients (Lin et al., 1995; Jeong et al., 1996), modeling and controlling the dynamic response of a plant (Ku et al., 1992; Han et al., 1996; Lin et al., 1996), and assessing risk probability (Hines, 1996). For surveys of applications in this industry, see also (Uhrig, 1991; Uhrig, 1994) (the above references are of course not exhaustive). The approach developped here for classifying reactor states is based on modeling the density of the data with a particular kind of mixture of Gaussians. The EM algorithm is a well-known learning algorithm that can be used to estimate the parameters of certain models, by iteratively (but efficiently) maximizing the likelihood of the data given the model and its parameters. In general, the learning data are unambiguously

labelled with the class to which they belong. Each class may then be modeled by one or more Gaussians (representing clusters of data points associated to that class). This is illustrated in Fig. 3.1(a). At the opposite, in absence of any information about the class of the learning data, we can interpret the relative contribution of each Gaussian (or cluster of Gaussians) in the mixture in order to estimate a posterior probability of a class that would be associated to the corresponding cluster(s). In this paper, our problem is intermediate: each time sequence is characteristic of some class but it may also contain "atypical subsequences" which resemble observations associated to the other classes. This is in part due to the fact that these time sequences cannot be easily segmented to represent strictly a particular reactor state. To take into account this particular prior knowledge on the data, we introduce here the notion of parameter sharing among Gaussian mixtures associated to different classes, as illustrated in Fig. 3.1(b). More precisely, starting with data clustered in "files" (each file being a multivariate time sequence for the data acquired during the a particular reactor operation), we will compute the parameters of the mixture associated to each file so that we can distinguish the typical data from spurious data that contaminate the file (and may be similar to data observed in the other files).

In section 2 we introduce notation and review basic notions on the application of the EM algorithm to Gaussian mixtures. In section 3 we present the Gaussian mixture model with shared Gaussian parameters, and we derive an EM algorithm for this model. In section 4 we present and discuss our experimental results on nuclear plant signals, and make a comparison with an approach purely based on signal classification using decision trees.

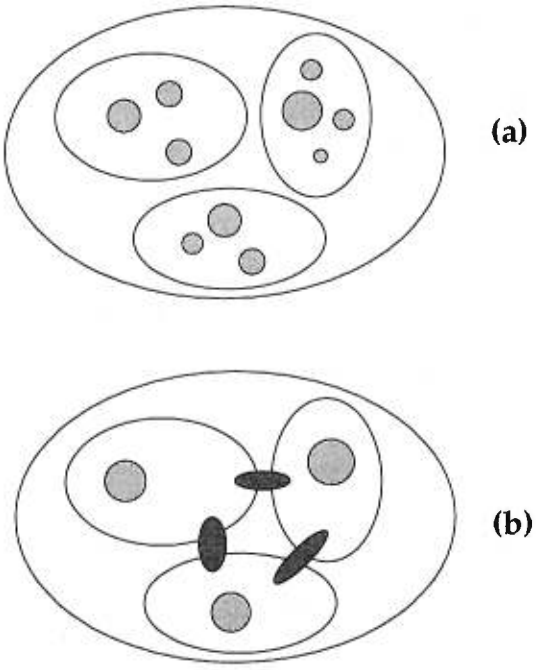

Figure 3.1.

(a) Mixture of mixtures of Gaussian densities. Each gray circle represents a Gaussian sub-class. (b) Some of the Gaussian components are shared across classes (darker grey).

3.3 The EM Algorithm for Gaussian Mixture Densities

3.3.1 The EM Algorithm

The EM (Expectation-Maximization) algorithm (Baum et al., 1970; Dempster et al., 1977) is a general optimization method that can be applied to certain maximum likelihood parameter estimation problems:

$$\theta^* = \text{argmax}_\theta \text{P}(\text{X} = \text{x}|\theta, \text{M}) \tag{3.1}$$

where θ is a vector of parameters, X is a random variable with observed value x, and M is the model. In many cases of interest the observation is actually a set of

independent and identically distributed observations:

$$X = X_1, X_2, X_3 ... X_T,$$

where T is the size of the training set. Therefore,

$$P(X = x) = \prod_{t=1}^{T} P(X_t = x_t).$$

In this paper we may sometimes write $P(x)$ instead of $P(X = x)$ (and similarly for other random variables).

The EM algorithm is based on a decomposition of the optimization problem obtained by introducing a (usually discrete) random variable in the model. The EM algorithm can generally be applied when the knowledge of this so-called *hidden variable* makes the optimization problem analytically feasible. For example, in the case of Gaussian mixtures, the hidden variable is the identity of the Gaussian (e.g., class) associated to each pattern. We will write Z_t for this class variable, for the t^{th} pattern. However, since this variable is generally not observed, the algorithm is based on the computation of the conditional *expected value* of the log-probability of the joint probability of Z and X: **E-Step**:

$$Q(\theta, \theta^k) = E_Z[\log(P(X, Z|\theta, M))|X = x, \theta^k] \tag{3.2}$$

where the expectation is taken over the values of the hidden variable Z, and where θ^k denotes the set of parameter values obtained at the k^{th} iteration of the EM algorithm. At each of these iterations, the parameters are updated as follows: **M-Step**:

$$\theta^{k+1} = \text{argmax}_\theta \text{Q}(\theta, \theta^{\text{k}}) \tag{3.3}$$

When this maximization can be done efficiently (e.g., when one can solve $\frac{\partial Q(\theta, \theta^k)}{\partial \theta} = 0$ for θ), the EM algorithm can be applied directly.

The seminal results of (Baum et al., 1970; Dempster et al., 1977) show that increasing this Q function is sufficient to increase the likelihood $P(x|\theta)$. More precisely,

$$Q(\theta^{k+1}, \theta^k) \geq Q(\theta^k, \theta^k) \rightarrow P(x|\theta^{k+1}) \geq P(x|\theta^k).$$

When Q is not fully maximized but only increased (e.g., with gradient-based optimization), one speaks of a Generalized EM algorithm.

3.3.2 Gaussian Mixtures

The EM algorithm can generally be applied to mixtures

$$P(X_t = x_t|\theta) = \sum_j P(X_t = x_t, Z_t = j|\theta) = \sum_j P(Z_t = j|\theta)P(X_t = x_t|Z_t = j, \theta)$$

where the $w_j = P(Z_t = j|\theta)$ are positive weights summing to one representing the prior probability of a point X_t being generated by one of the models $P(X_t|Z_t = j, \theta)$.

In the case of Gaussian mixtures,

$$P(x_t|j, \theta) = \frac{\exp(-0.5(x_t - \mu_j)'\Sigma_j^{-1}(x_t - \mu_j))}{(2\pi)^{\frac{n}{2}}\sqrt{|\Sigma_j|}}$$

where $\theta = \{w, \theta_1, \theta_2, \ldots, \theta_K\}$ is a set of parameters including K parameter sets for each Gaussian component parametrized by $\theta_j = (\mu_j, \Sigma_j)$ ($n \times 1$ mean vector and $n \times n$ covariance matrix), x' is the transpose of the vector x, and $|\Sigma|$ denotes the determinant of the matrix Σ. In simpler models, the covariance matrix can be forced to be diagonal or a multiple of the identity matrix.

Solving the M-Step for Gaussian mixtures yields to the following re-estimation formulae for each iteration of the EM algorithm. To lighten the notation, we will introduce the notion of *posteriors*

$$p_{j,t,k} \stackrel{\text{def}}{=} P(Z_t = j|x_t, \theta^k) = \frac{\omega_j P(x_t|j, \theta^k)}{P(X_t = x_t|\theta^k)},$$

using the parameters estimated at the k^{th} iteration of the algorithm. For **mixture weights**:

$$w_j^{k+1} \leftarrow \frac{\sum_{t=1}^T p_{j,t,k}}{T} \tag{3.4}$$

For **Gaussian means**:

$$\mu_j^{k+1} \leftarrow \frac{\sum_{t=1}^T p_{j,t,k} x_t}{\sum_{t=1}^T p_{j,t,k}}$$

For **Gaussian covariance matrices**:

$$\Sigma_j^{k+1} \leftarrow \frac{\sum_{t=1}^{T} p_{j,t,k} x_t x_t'}{\sum_{t=1}^{T} p_{j,t,k}} - \mu_j^{k+1} \mu_j'^{k+1}$$

In the next section, we will show how similar (but different) re-estimation formulae arise when the model is a mixture of mixtures of Gaussians whose parameters may be shared.

3.4 An EM Algorithm for Shared Gaussian Mixtures

For the purpose of taking advantage of prior knowledge that is specific to our application we present here a special case of the EM algorithm applied to Gaussian mixtures with shared Gaussian parameters.

3.4.1 The Model

The training data has been partitionned into K classes (K is 4 in our application). Furthermore, it has been assumed that within each class certain sub-classes may exist which resemble each of the other classes. One of the objectives of this model is to assign new data points to one of these classes or sub-classes, or to identify them as "abnormal" data points which do not belong to any of the known regimes of the nuclear plant.

At the highest level, we will represent the model as a mixture over the main classes. These high-level classes correspond to sensor log *files*, hence we introduce the random variable F_t to denote the high-level class associated to the t^{th} sensor pattern. F_t takes values $f = 1 \ldots K$. The high-level mixture is thus expressed:

$$P(X_t = x_t|\theta) = \sum_{f=1}^{K} \alpha_f \, P(X_t = x_t|F_t = f_t, \theta) \tag{3.5}$$

where $\alpha_f = P(F_t = f)$ is the prior probability of finding an observation x_t in the f^{th} class. Note that in our application F_t is observed *for the training data* (but not for

new, test data points), so training will be based on maximizing

$$P(X|F,\theta) = \prod_{t=1}^{T} P(X_t = x_t|F_t = f_t, \theta)$$

using a training set $\{(x_1, f_1), (x_2, f_2), \ldots, (x_T, f_T)\}$.

Each of these class models is itself seen as a mixture of K Gaussian components (corresponding to the sub-classes). One of these components is specific to that class while the $K-1$ other components *are shared* with the $K-1$ other classes: the i^{th} Gaussian component of the f^{th} class has the same parameters as the f^{th} Gaussian component of the i^{th} class. To formalize this we introduce another state variable, J_t with values $j = 1 \ldots K$, representing the identity of the sub-class (i.e. of the Gaussian) associated to the t^{th} pattern within the file F_t:

$$P(X_t = x_t|F_t = f, \theta) = \sum_{j=1}^{K} w_{fj} P(X_t = x_t|F_t = f, J_t = j, \theta_{fj})$$

in which $w_{jf} = P(J_t = j|F_t = f)$ is the prior probability of the j^{th} Gaussian within the f^{th} model. The sharing across sub-models is obtained with the constraints

$$\theta_{fj} = \theta_{jf} \tag{3.6}$$

where $\theta_{fj} = (\mu_{fj}, \Sigma_{fj})$, and $P(X_t = x_t|F_t = f, J_t = j, \theta_{fj})$ is a Gaussian distribution with mean μ_{jf} and covariance matrix Σ_{fj}. Note that the parameters $\theta_{ff} = (\mu_{ff}, \Sigma_{ff})$ represent a Gaussian distribution that is specific to the f^{th} file, whereas θ_{fj} for $f \neq j$ represents a sub-class that is observed both in the f^{th} file and the j^{th} file.

To summarize, the whole set of parameters is

$$\theta = (\{\alpha_f\}, \{w_{fj}\}, \mu_{fj}, \Sigma_{fj})$$

with f and j ranging from 1 to K, equality constraints in equation 3.6, and probabilistic constraints

$$\alpha_f \geq 0, \quad \sum_f \alpha_f = 1$$

and

$$w_{fj} \geq 0, \ \sum_j w_{fj} = 1.$$

3.4.2 Deriving the Re-Estimation Formulae

To derive the EM re-estimation formulae for this particular type of mixture model, we will write down the $Q(\theta, \theta^k)$ auxiliary function (E-step) and solve for its maximum (M-step).

Since the file (super-class) F_t associated to each training pattern is observed in our case, we can easily estimate α_i, and we will only need to consider the maximization of the conditional likelihood $P(X|F, \theta)$. By marginalizing the joint observations (x_t, f_t) and counting, we obtain the estimation of the marginal probabilities

$$\alpha_f = P(F = f) = \frac{\sum_{t=1}^{T} I_{f_t=f}}{T}$$

where I_e is the indicator function, equal to 1 when e is true and 0 otherwise.

Let us now consider the more difficult task of estimating the sub-class weights w_{fj} and Gaussian parameters θ_{fj}. Applying equation 3.2 to the model described in the previous sub-section, and adding Lagrange multipliers to represent the constraints on w_{fj}, we obtain:

$$\begin{aligned} Q(\theta, \theta^k) &= E_J[\log P(X, J|F, \theta)|X, F, \theta^k] + \text{constraints} \\ &= \sum_t \sum_j P(J_t = j|f_t, x_t, \theta^k) \log P(x_t, J_t = j|f_t, \theta) + \sum_f \lambda_f (1 - \sum_j w_{fj}) \\ &= \sum_t \sum_j p_{j,t,k} (\log w_{f_t,j} + \log P(x_t|J_t = j, f_t, \theta)) + \sum_f \lambda_f (1 - \sum_j w_{fj}) \end{aligned}$$

where $w_{f_t,j} = P(J_t = j|F_t = f_t)$, and where the posteriors on individual sub-class Gaussians have been denoted

$$p_{j,t,k} = P(J_t = j|F_t = f_t, X_t = x_t, \theta^k) \tag{3.7}$$

at the k^{th} iteration of the EM algorithm.

We will now separately solve the equations $\frac{\partial Q(\theta,\theta')}{\partial \theta} = 0$ for the sub-mixture weights w_{fj} and for the Gaussian parameters μ_{fj} and Σ_{fj}.

For the sub-mixture weights, we obtain

$$\frac{\partial Q(\theta, \theta^k)}{\partial w_{fj}} = \frac{\sum_t I_{f_t=f}\, p_{j,t,k}}{w_{fj}} - \lambda_f = 0$$

which yields to

$$w_{fj} = \frac{\sum_t I_{f_t=f}\, p_{j,t,k}}{\lambda_f}.$$

From the constraint on the sum of w_{fj}, we finally obtain, for the $(k+1)^{\text{th}}$ iteration of the EM algorithm:

$$w_{fj}^{k+1} = \frac{\sum_t I_{f_t=f}\, p_{j,t,k}}{\sum_j \sum_t I_{f_t=f}\, p_{j,t,k}} \tag{3.8}$$

Note that this formula is essentially the same as the one for ordinary Gaussian mixtures (equation 3.4).

Let us now apply a similar procedure to the re-estimation of the means, with particular attention to the sharing constraint ($\mu_{fj} = \mu_{jf}$):

$$\frac{\partial Q(\theta, \theta^k)}{\partial \mu_{fj}} = \sum_t (I_{f_t=f}\, p_{j,t,k} \Sigma_{fj}^{-1}(x_t - \mu_{fj}) + I_{f_t=j}\, p_{f,t,k} \Sigma_{fj}^{-1}(x_t - \mu_{fj})) = 0$$

By pre-multiplying by Σ_{fj} and then isolating the vector μ_{fj} we finally obtain for the $(k+1)^{\text{th}}$ iteration of the EM algorithm,

$$\mu_{fj}^{k+1} = \frac{\sum_t (I_{f_t=f}\, p_{j,t,k} \;+\; I_{f_t=j}\, p_{f,t,k}) x_t}{\sum_t (I_{f_t=f}\, p_{j,t,k} \;+\; I_{f_t=j}\, p_{f,t,k})} \tag{3.9}$$

Similarly, one can obtain the re-estimation formula for the covariance matrices:

$$\Sigma_{fj}^{k+1} = \frac{\sum_t (I_{f_t=f}\, p_{j,t,k} \;+\; I_{f_t=j}\, p_{f,t,k})\; x_t x_t'}{\sum_t (I_{f_t=f}\, p_{j,t,k} \;+\; I_{f_t=j}\, p_{f,t,k})} \;-\; \mu_{fj}^{k+1} \mu_{fj}^{\prime k+1} \tag{3.10}$$

3.4.3 Parameter Initialization

It is well known that the EM algorithm for Gaussian mixtures can get stuck in local maxima of the likelihood (in fact, unless a penalty on small variances is included, the

global maximum is a very poor solution with one of the Gaussians on a single point and zero variance, which gives infinite likelihood). For these reasons, the parameters initialization is important. For the proposed model, we will take advantage of our prior knowledge on the data and the interpretation of the model in order to initialize the Gaussian parameters.

The basic idea is to initialize the Gaussian parameters for the Gaussian component that is specific to a file f by fitting a Gaussian to all of the data from the file f:

$$\begin{aligned} \mu^0_{ff} &= \frac{\sum_t I_{F_t=f}\, x_t}{N_f} \\ \Sigma^0_{ff} &= \frac{\sum_t I_{F_t=f}\, x_t x_t'}{N_f} - \mu_{ff}\mu'_{ff} \end{aligned} \tag{3.11}$$

where $N_f = \sum_t I_{F_t=f}$ is the number of points in the f^{th} file.

Similarly, the Gaussian parameters associated to files i and j will be initialized by fitting a Gaussian to the data from the two files i and j:

$$\begin{aligned} \mu^0_{ij} &= \frac{\sum_t (I_{F_t=i} + I_{F_t=j})\, x_t}{N_i + N_j} \\ \Sigma^0_{ij} &= \frac{\sum_t (I_{F_t=i} + I_{F_t=j})\, x_t x_t'}{N_i + N_j} - \mu_{ij}\mu'_{ij} \end{aligned} \tag{3.12}$$

As described in the next sub-section, experiments performed with artificial data showed that this initialization procedure helps to find good initial solutions and avoid local maxima of the likelihood.

3.4.4 Validation on Artificial Data

To empirically verify the correctness of our algorithm and our implementation we tested the EM re-estimation formula and the above initialization procedure on a simple case with 2 variates and 2 files, i.e., 3 sets of Gaussian parameters (θ_{11}, θ_{12}, and θ_{22}). There are 200 data points in each file, and in each file data points are

generated half of the time from the file-specific Gaussian and half of the time from the shared Gaussian.

The data set and the trajectories of the 3 means are illustrated in figure 3.2, when the parameters are initialized as described in the previous sub-section. The figure shows how the initialization procedure separates the Gaussians in a sensible way. The trajectories of the Gaussian means start near the origin and gradually move towards the cluster centers. The points in the top cluster are generated by the shared Gaussian (100 points in each file). The points in the bottom left and right parts of the figure are respectively generated by the first and second file-specific Gaussians.

In contrast, figure 3.3 illustrates what could happen when the initialization procedure does not take into account the structure of the model, and the EM algorithm gets stuck in a poor local maximum of the likelihood function. To really understand what goes wrong in this case one has to know that the Gaussian which stayed stuck in the middle is the shared Gaussian. If it moved to the bottom left, the likelihood of the top points in the second file would drop very low. But on the first file the specific Gaussian holds the ground of the top cluster and prevents the shared Gaussian from moving there.

3.5 Experiments

3.5.1 Data Set

The shared Gaussian mixture model was tested on a data set collected at the Pointe-Lepreau nuclear plant in New-Brunswick, Canada. The data was extracted at a 6 second sampling rate over 5 periods of $33\frac{1}{3}$ hours each, to obtain 20000 data points for each of 5 regimes of the nuclear plant. The data was further subsampled 10-fold before further processing because of the continuity of the time-series. The sensors measure thermohydraulic and power properties of the reactor: inlet and oulet flows, temperatures and pressures and pressurizer caracteristics. The measurements from

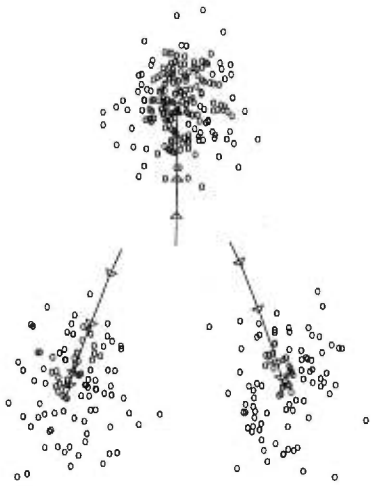

Figure 3.2.

Training data and evolution of Gaussian means for the test with artificial data points, using the proposed initialization procedure. Each line segment with an arrow represents the movement of a mean for one EM iteration.

7 different sensors were collected at each time step. From visual inspection, the first three sensors were believed to be most important, and the experiments with the decision tree confirmed that (since they were the only ones used in the decision nodes).

For some of the regimes, the data was not obtained in a continuous time sequence but was obtained in multiple measurements. This particularity was used to evaluate on periods not used for training the generalization of a model trained with the data from only some of the measurement periods. An example of a data sequence (for the first three sensors) is shown in Figure 3.4.

The 4 types of regimes in which the data was collected are the following:

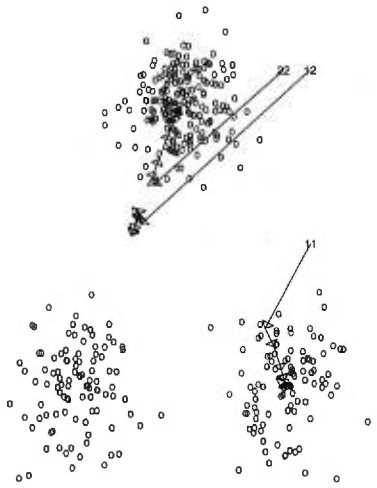

Figure 3.3.

Training data and evolution of Gaussian means for the test with artificial data points, using an initialization which yields to a poor local maximum of the likelihood.

1. Normal plant operation (N).

2. Refueling (R).

3. Plant operating at low power (L).

4. Governor valve testing, hereafter named govalve (G).

A test set was constituted using a few hundred data points in an interval that was not in the training set.

The software for the experiments described in this paper was written using the S-Plus language and statistical library.

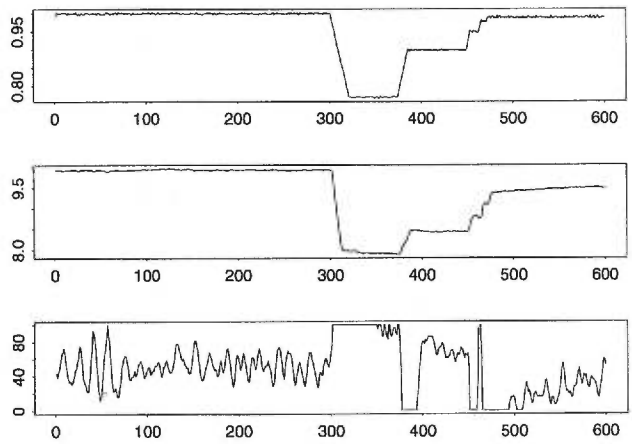

Figure 3.4.

An example of signal containing a refueling event. This sequence is not part of the training set but was used to test the Gaussian mixtures. The refueling event is around time steps 300.

3.5.2 Application of a Decision Tree Model

Before exploring the application of the shared Gaussian mixture model, we studied the application of decision tree models for classification (CART) to this data. The study of the results of these experiments was one of the motivations for designing the shared Gaussian mixture model.

A classification tree is the result of a recursive binary partition of the set of observations. Each binary partition defines a node and splits the data into the data associated to each of its children. For classification trees the partitioning function is chosen using heuristics that attempt to make the class distributions in each of the children more homogeneous. More specifically in our experiments we used the entropy of the class distribution as a measure of homogeneity. The partitioning function splits the data according to a threshold value for one of the input variables.

A greedy algorithm builds the tree recursively by choosing for each node the input variable and the threshold value which optimize the entropy of the children with respect to the entropy of the parent node. The leaves of the tree are associated to a constant decision rule that simply chooses the most frequent class among the patterns that fall in that leaf. After the training set is learned perfectly, a pruning algorithm reduces it using 10-fold cross-validation and minimizing a cost-complexity measure combining the number of errors and the number of leaves (i.e., the complexity of the tree).

The training set had seven input variables and one target class variable, corresponding to one of the 4 plant regimes described earlier. The initial tree (before pruning) had 33 leaves, and it was reduced to 9 leaves with pruning. The tree is represented in Figure 3.5.

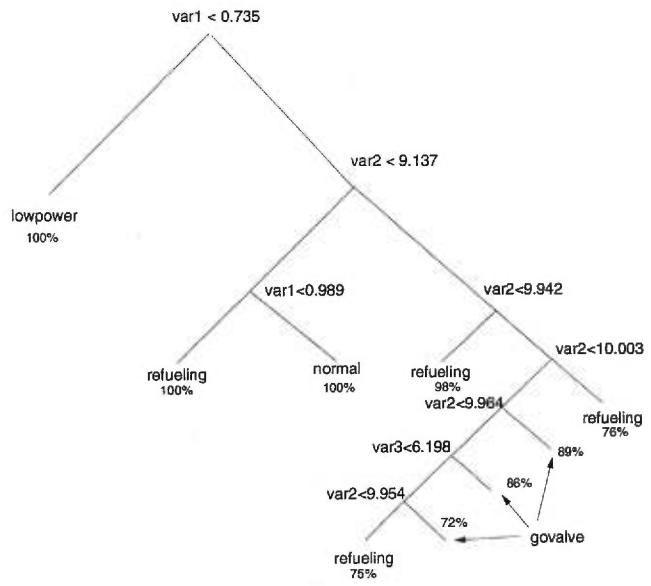

Figure 3.5.

Decision tree obtained with the CART algorithm using the seven sensors. Note that only three variables are used in the decision nodes.

An interesting result from this experiment is that the decision tree helps us identify variables which are most important in order to discriminate among the classes. We used these variables (the first three sensor readings) in the Gaussian mixtures experiments. Among these three variables, variable 2 appears to be the most important (it is used in 5 of the 8 decision nodes). One problem with the decision tree approach, of course, is that it only models the decision boundary between classes and does not allow us to evaluate whether a new sequence is similar to the data that has already been seen, i.e., it can't be used to identify potentially abnormal situations.

3.5.3 Experimental Results with the Shared Mixtures Model

Here, in order to evaluate the behavior of the model, we consider time sequences that have not been used in training it. Such a model can be used to diagnose the state

of the reactor in real time or to identify the state described by some windowed time series of the data (batch mode). For the present work, we investigate the "on-line" responses of the mixture model fed with unlearned data.

For each data point x, we study the response of the mixture model with the component posteriors

$$p_{i,j} = \frac{\omega_{i,j} P(X = x|i, \theta_j)}{\sum_l \omega_{i,l} P(X = x|i, \theta_l)}. \tag{3.13}$$

To take into account the fact that some components share parameters, we defined "shared posteriors" as follows:

$$\Pi_{i,j} = p_{i,j} + p_{j,i}, \quad \text{if} \quad i \neq j \tag{3.14}$$

$$\Pi_{i,i} = p_{i,i} \tag{3.15}$$

Each data point can then be classified according to an index pair

$$(i, j) = \text{argmax}_{(i',j')} \Pi_{i',j'}$$

These indices can be interpreted as follows.The first index defines a "regime" in the same way it was defining a specific time series (a data file) during the training. The second index gives information about the "state" of the system within that regime. Because of the way the parameters are shared, the index pair (i, j) is equivalent to the index pair (j, i). The roles played by the two indices are complementary: for some particular regime, we expect to detect different states that describe the sucessive stages of the on-going regime.

For the sake of illustration, we consider a model trained with 20 iterations of EM and for which the likelihood has been saturated during the learning. See Fig. 3.6 for the evolution of likelihood during training.

Testing on Known Classes

We will now show results obtained when testing this model on the out-of-sample data from signal classes that have been seen during training.

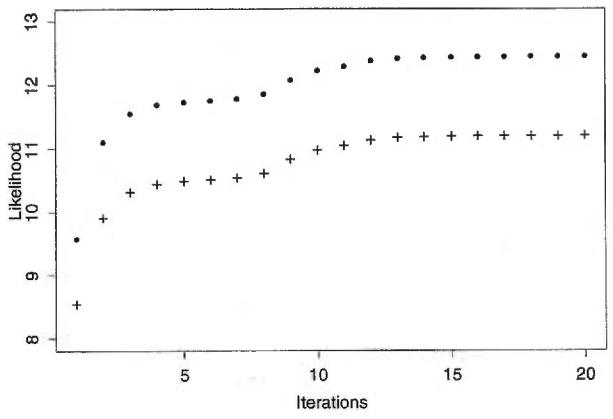

Figure 3.6.

Evolution of the likelihood of the training data during training of the shared Gaussian mixture model. The lower curve is the likelihood on the test data.

First, we examine the three most significant shared posteriors during a govalve (G) session, in Fig. 3.7. The other shared posteriors have near-zero values during that time sequence. Note how the govalve class is clearly identified during this test sequence.

Next, we consider a more complicated sequence, corresponding to the input data displayed in Fig. 3.4, that is from a refueling session. The values of the four strongest shared posteriors for this sequence are shown in Fig. 3.8. This curve shows that during this sequence, one refueling event occured that is preceded by govalve event. The top three curves represent refueling sub-classes and the bottom one represents the typical govalve sub-class. After obtaining those curves, we learned from the plant engineers that such a sequence of events (govalve preceding refueling in a refueling sequence) is typical in the plant operation.

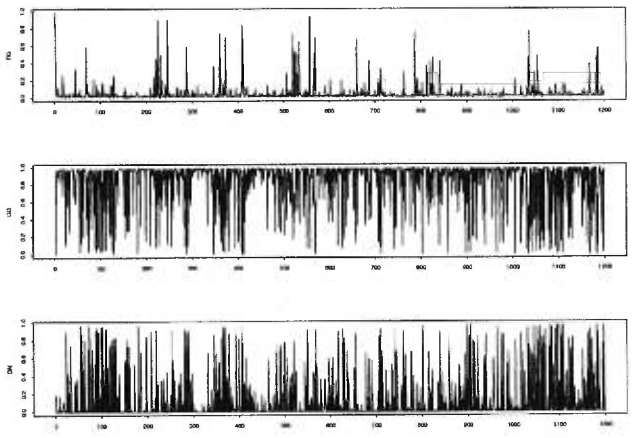

Figure 3.7.

Sequence of shared posterior values during an out-of-sample govalve session. For example the curve labeled RG corresponds to the sub-class "Refueling-Govalve".

Testing on Unknown Classes

Finally, we consider what happens when the model trained on the four classes (Refueling, Govalve, Normal, Low power) is tested on data acquired while the plant was behaving in a different mode. This out-of-sample and "out-of-class" data was collected during the "run-up" mode of operation of the reactor. We use this data to verify if the model properly identifies this regime as unknown. We finally compare the behavior of the model on three test sequences: the govalve and refueling test sequences discussed already, and the "run-up" test sequence.

For this purpose we have computed the data likelihood for each point of the three test sequences, as defined in equation 3.5. The likelihood of each of the test points for these three sequences is shown in Fig. 3.9. As can clearly be seen, the model responds strongly and consistently only for the known classes (top two curves) whereas the likelihood of the unknown class observations is comparatively very low.

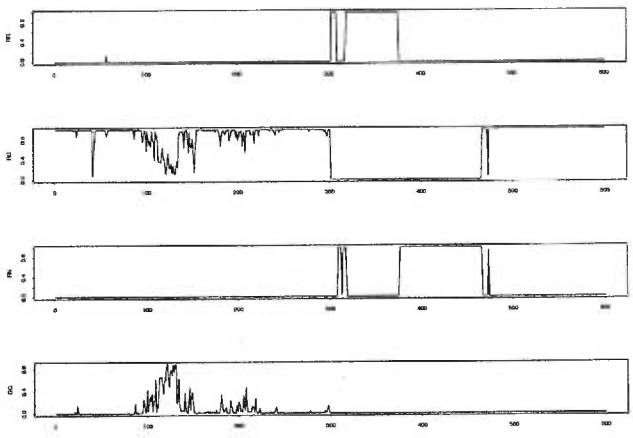

Figure 3.8.

Sequence of shared posterior values during a refueling session. For example the curve labeled RR corresponds to the sub-class "Refueling-Refueling".

3.6 Conclusion

In this paper we have introduced a new learning model based on shared Gaussian mixtures, and applied it successfully to the monitoring and classification of nuclear reactor signals.

The shared Gaussian mixtures model takes advantage of prior knowledge on that type of signal, in which within each type of plant operation sequence, some subsequences similar to other plant operations may occur. An EM algorithm for estimating the parameters of the model was presented, which allows quick convergence to a solution during learning.

The model was trained on four classes of plant behaviors and tested on out-of-sample data from the known classes as well as unknown classes, showing that the model can be used to analyze the behavior of the plant (classification among known behaviors) as well as identify unknown, potentially abnormal behaviors, which is

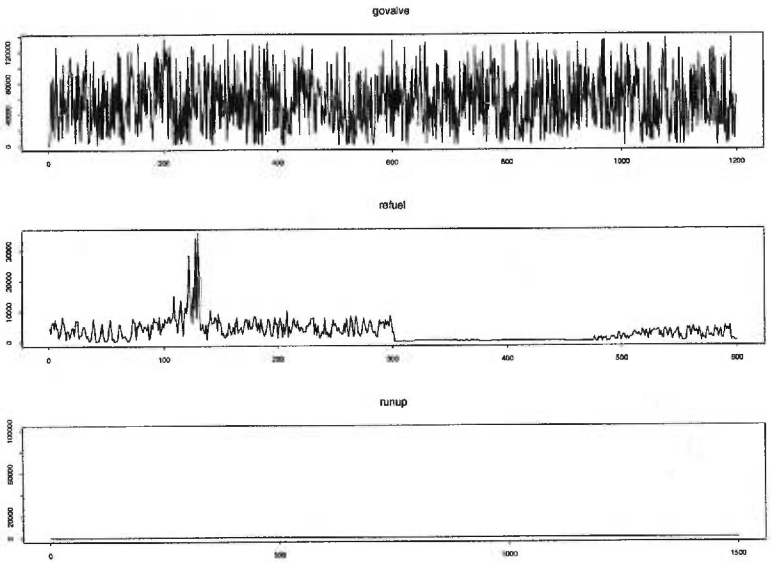

Figure 3.9.

Sequence of likelihoods for each data point in the three test sequences.Top two sequences are from known classes. Bottom sequence is for a unknown class.

critical for plant safety.

Acknowledgements

This work is supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada. We express our appreciations to N.B. Power for making data available for testing.

Chapitre 4

TROISIÈME ARTICLE: TESTS D'HYPOTHÈSES À L'AIDE D'UNE STATISTIQUE HORS-ÉCHANTILLON

Dans cet article, nous proposons une méthode pour nous aider à répondre à la question suivante: sommes-nous en mesure de faire des prédictions à l'aide du modèle choisi? Afin de répondre à cette question, nous présentons une démarche qui repose sur très peu d'hypothèses distributionnelles.

Une approche conventionelle consiste à supposer une certaine relation de dépendance entre la variable à prévoir et des variables prédictives, d'estimer les paramètres de la relation et de tester si les paramètres inférés diffèrent significativement d'une valeur associée à une hypothèse nulle. Des problèmes peuvent survenir avec une telle approche. Dans des cas complexes, comme celui des rendements d'actifs financiers que nous considérons dans cet article, nous avons peu d'assurance vis-à-vis une forme fonctionnelle particulière. Donc, utiliser une méthode qui est d'abord conçue pour fournir un estimé non biaisé de la valeur des vrais paramètres peut être discutable.

Nous proposons d'utiliser, comme dans les précédents articles, les erreurs de généralisation, c'est-à-dire les erreurs mesurées sur des données qui n'ont pas été utilisées pour construire et choisir le modèle, afin de répondre à la question énoncée plus haut. Nous présentons un algorithme qui nous fournit un estimé non biaisé de l'erreur de généralisation applicable à des séries chronologiques (où l'ordre des données est important). Nous discutons des conclusions obtenues par l'application de l'inférence traditionnelle (utilisant tout l'échantillon pour faire les tests) et par l'utilisation de statistiques hors-échantillon. Nous comparons notre approche à un modèle où les corrections à apporter aux statistiques mesurées sont connues, c'est-à-dire où nous pouvons construire un écart-type de la distribution de la statistique qui

tient compte des autocorrélations des résidus. Les corrections apportées reposent sur quelques hypothèses dont nous serons amenés à discuter la justesse face aux données financières utilisées. Notre approche, ne reposant pas sur des hypothèses distributionnelles a l'avantage de s'étendre à n'importe quel modèle, pour lequel la distribution des statistiques estimées sur tout l'échantillon est inconnue.

Nous présentons aussi deux tests portant sur des hypothèses nulles légèrement différentes: 1) pas de relation entre les entrées et les sorties et 2) est-ce que les entrées nous permettent d'obtenir des prédictions meilleures que celles fournies par un modèle naïf? Les tests conduits dans le cas de la première hypothèse nous permettent de présenter différentes méthodes reposant sur le "bootstrap" ainsi que d'estimer des quantités qui s'avèreront utiles pour conduire les tests sur la seconde hypothèse. Les résultats des tests faits sur la deuxième hypothèse nous disent que, même si nous pouvons rejeter l'hypothèse *pas de relation entre les entrées et les sorties*, nous sommes loin de pouvoir rejeter l'hypothèse *pas de prévision possible en utilisant les entrées*, étant donné une classe de fonctions. Ce résultat illustre l'importance de considérer l'hypothèse qui traduit le mieux ce qui nous importe de tester (la prévision). Tester une hypothèse apparemment similaire (pas de dépendance) peut conduire à des conclusions très différentes.

Les résultats obtenus indiquent quelques pistes de développement que nous exposerons en conclusion de cette thèse. Ces travaux sont disponibles en prépublication du CRM sous le numéro CRM-2585.

4.1 L'hypothèse de retour à la moyenne

L'hypothèse de retour à la moyenne concernant les prix des actifs boursiers est monnaie courante. Les tenants de cette hypothèse soutiennent qu'il existe une valeur d'équilibre pour les prix et que les déviations des prix observés sur le marché par rapport à cette valeur d'équilibre seront corrigées. Cette hypothèse est reliée aux hy-

pothèses d'efficacité des marchés et de constance dans le rendement requis pour qu'un investisseur accepte de prendre des risques financiers. Une conséquence de l'hypothèse du retour à la moyenne est la présence d'autocorrélations négatives dans les séries de prix des actifs. Par contre, l'autocorrélation négative dans les séries de prix peut aussi avoir d'autres explications (Campbell et al., 1997; Poterba and Summers, 1988). Un des premiers résultats, qui continue à alimenter le débat autour de cette hypothèse, est exposé dans (Fama and French, 1988). Différents auteurs ont observé, depuis, la présence d'autocorrélations négatives dans les prix[1], tandis que d'autres présentent des résultats qui laissent planer un sérieux doute sur la véracité de l'hypothèse du retour à la moyenne. Différents résultats sont discutés dans (Campbell et al., 1997). Un modèle de prix d'actifs boursiers expliquant la présence d'autocorrélations négatives est celui qui propose de représenter par deux composantes les prix: une composante stationnaire et une composante aléatoire. La variance d'un processus aléatoire augmente proportionnellement avec l'horizon, ou l'échelle, à laquelle nous l'observons. Le comportement de la variance de la composante stationnaire dépend du modèle utilisé (Campbell et al., 1997; Fama and French, 1988). Fama et French montrent que la présence de telles composantes fera en sorte que la régression des rendements des actifs pour différents horizons donnera lieu à une courbe en forme de U pour la pente de la régression (et pour le coefficient d'autocorrélation) des rendements. Cette courbe devrait être près de zéro pour des horizons courts, de plus en plus négative avec l'accroissement de l'horizon, puis remontera vers zéro par la suite. La remontée vers zéro serait causée par la dominance de la variance du bruit blanc à mesure que l'horizon s'accroît.

Une des causes du débat peut se trouver dans les différentes hypotèses qui sont faites lors de la construction des tests statistiques. Il y a notamment de l'autocorrélation dans les rendements utilisés, et cette autocorrélation augmente avec l'horizon. Notre

[1] D'autres tests, fondés sur des statistiques différentes semblent, parfois, aller dans le sens de l'hypotèse de retour à la moyenne.

but n'est pas ici de trancher le débat, mais de développer des tests qui ne reposent pas sur des hypothèses distributionnelles ou des résultats asymptotiques. Nous utilisons le modèle de Fama et French (Fama and French, 1988) pour exposer notre approche.

On Out-of-Sample Statistics for Financial Time-Series

François GINGRAS
Département de Physique
Université de Montréal

Yoshua BENGIO
Laboratoire d'Informatique des Systèmes Adaptatifs
Département d'Informatique et Recherche Opérationnelle
Université de Montréal

Claude NADEAU
Centre Interuniversitaire de Recherche en ANalyse des Organisations

This paper studies an out-of-sample statistic for time-series prediction that is analogous to the widely used R^2 in-sample statistic. We propose and study methods to estimate the variance of this out-of-sample statistic. We suggest that the out-of-sample statistic is more robust to distributional and asymptotic assumptions behind many tests for in-sample statistics. Furthermore we argue that it may be more important in some cases to choose a model that generalizes as well as possible rather than choose the parameters that are closest to the true parameters. Comparative experiments are performed on a financial time-series (daily and monthly returns of the TSE300 index). The experiments are performed for varying prediction horizons and we study the relation between predictibility (out-of-sample R^2), variability of the out-of-sample R^2 statistic, and the prediction horizon.

4.2 Introduction

The purpose of the analysis of time-series such as financial time-series is often to take decisions based on data $D_T = \{z_1, \ldots, z_T\}$, with $Z_t = (X_t, Y_t)$. In this paper, we will focus on decisions which take the form of a prediction $\hat{y}_{T+h}$ of the future value of some variable [2], say Y_{T+h}. The quality of the prediction will be judged a posteriori according to some loss function, such as the squared difference between the prediction $\hat{y}_{T+h}$ and the realization Y_{T+h} of the predicted variable $(\hat{y}_{T+h} - Y_{T+h})^2$. A common approach is to use the historical data D_T to infer a function f that takes as input the value of some summarizing information X_t and produces as output $\hat{y}_t = f(X_t)$, which in the case of the above quadratic loss function would be an estimate of the conditional expectation $E[Y_t|X_t]$. The hope is that if this function worked well on observed past pairs (x_t, y_t), it should work well on (X_{T+h}, Y_{T+h}) [3].

How should we choose the function f? A classical approach is to assume a parametrized class of functions, like affine functions, estimate the value of these parameters by maximum likelihood or least squares. Then the model is accessed via goodness-of-fit tests and statistical tests to verify if these parameters differ significantly from the value that would be consistent with a null hypothesis (e.g., the parameters of the regression are significantly different from zero, so that there is really a linear dependency between the X's and the Y's). In particular, these tests are important to know whether one should use the proposed model at all, or to decide among several models.

In this paper we will consider alternative approaches to address the last question, i.e., how a model should be validated and how several models should be compared.

---

[2] In this paper we will normally use upper case for random variables and lower case for their value.

[3] Obviously $\hat{y}_{T+h} = f(X_{T+h})$ is computable only if X_{T+h} is available. We will typically consider lagged variables so that X_{T+h} is available at "time" T if X is lagged by an amount greater or equal to h.

It is very satisfying to obtain a result on the **true value** of the parameters (e.g., to use an efficient estimator, which converges as fast as possible to the true value of the parameters). But in many applications of time-series analysis, the end-user of the analysis may be more interested in knowing whether the model is going to work well, i.e., to generalize well to the future cases. In fact, we will argue that sometimes (especially when data is scarce), the two objectives (estimating the true parameters or choosing the model that generalizes better) may yield very different results. Another fundamental justification for the approach that we are putting forward is that we may not be sure that the true distribution of the data has the form (e.g. linear, Gaussian, etc...) that has been assumed. Therefore it may not be meaningful to talk about the true value of the parameters, in this case. What may be more appropriate is the question of generalization performance: will the model yield good predictions in the future? where the notion of "good" can be used to compare two models. To obtain answers to such questions, we will consider statistics that measure **out-of-sample performance**, i.e., measured on data that was not used to form the prediction function. This contrast with the in-sample R^2 used in predictability tests (Campbell et al., 1997; Kaul, 1996).

Using a measure of performance based on out-of-sample errors is an approach gaining in popularity. In econometrics, Diebold and Mariano (Diebold and Mariano, 1995) are using out-of-sample errors (or predictive performances) to build tests on accuracy measures. In the machine learning community, it is common to use such measures of performance. Splitting the data into a training subset and a test subset is a popular option. For smaller data sets, K-fold cross-validation is preferred (Efron and Tibshirani, 1993). The above methods may not be applicable to sequential data, and in the non-stationary case may yield optimistic estimates.A more honest estimate can be obtained with a **sequential cross-validation procedure**, described in this paper. This estimate essentially attempts to measure the predictability of the time-series when a particular class of models is used. In this context, what the analyst will

try to choose is not just a **function** $f(x_t)$, but a **functional** F that maps historical data D_t into such a function (and will be applied to many consecutive time steps, as more historical data is gathered).

The objective of this paper is three-fold. First, establish a distinction between two apparently close null hypotheses: (1) no relationship between the inputs and the outputs and (2) no better predictive power of a given model with respect to a naive model. Second, we propose methods to test the second null hypothesis. Third, we show that these two types of tests yield very different results on commonly studied financial returns data.

In section 4.3, we present the classical notion of generalization error, empirical risk minimization, and cross-validation, and we extend these notions to sequential data. We also present the notion of a "naive model" used to establish a comparison benchmark (and null hypotheses).

In section 4.4, we introduce a measure of forecastibility, R_o, that is related to the one defined by Granger and Newbold (Granger and Newbold, 1976). Its estimator, $\hat{R}_o$, is presented.

Section 4.5 describes the financial time-series data and presents some preliminary results.

In section 4.6, we test the hypothesis of non-relation between the inputs and the outputs. Although this hypothesis is not really what we want to test, it allows us to nicely introduce some difficult issues with the data at hand, such as dependency induced by overlapping, and the type of methodologies used later on, including the bootstrap. Furthermore, we will compare the results obtained on that test and the test concerning generalization error. Concerning the no-dependency test, we perform a simulation study to compare the power of in-sample and out-of-sample statistics.

Section 4.7 aims at assessing whether inputs may be used to produce forecasts that would outperform a naive forecast. Following section 4.4, we test if $R_o = 0$ against the alternative that it is positive. We do so for different prediction horizons,

using the statistic $\hat{R}_0$ and various bootstrap schemes. The results are compared to those obtained when trying to reject the null hypothesis of no dependency, allowing us to show a notable distinction between the absence of relationship between inputs and outputs and the inability of inputs to forecast outputs.

We conclude the paper with a discussion of the results in section 4.8.

4.3 Expected Risk and Sequential Validation

This section reviews notions from the generalization theory of Vapnik (Vapnik, 1995), and it presents an extension to sequential data of the concepts of generalization error and cross-validation. We also define a "naive" model that will be used as a reference for the R_o statistic.

First let us consider the usual i.i.d. case (Vapnik, 1995). Let $Z = (X, Y)$ be a random variable with an unknown density $P(Z)$, and let the *training set* D_l be a set of l examples $z_1, \ldots, z_l$ drawn independently from this distribution. In our case, we will suppose that $X \in \mathcal{R}^n$ and $Y \in \mathcal{R}$. Let $\mathcal{F}$ be a set of functions from $\mathcal{R}^n$ to $\mathcal{R}$. A measure of loss is defined which specifies how well a particular function $f \in \mathcal{F}$ performs the generalization task for a particular Z: $Q(f, Z)$ is a functional from $\mathcal{F} \times \mathcal{R}^{n+1}$ to $\mathcal{R}$. For example, in this paper we will use the quadratic error $Q(f, Z) = (Y - f(X))^2$. The objective is to find a function $f \in \mathcal{F}$ that minimizes the expectation of the loss $Q(f, Z)$, that is the **generalization error** of f:

$$G(f) = E[Q(f, Z)] = \int Q(f, z)P(z)dz \tag{4.1}$$

Since the density $P(z)$ is unknown, we can't measure or even less minimize $G(f)$, but we can minimize the corresponding **empirical error**:

$$G_{emp}(f, D_l) = \frac{1}{l} \sum_{z_i \in D_l} Q(f, z_i) = \frac{1}{l} \sum_{i=1}^{l} Q(f, z_i). \tag{4.2}$$

When f is chosen independently of D_l, this is an unbiased estimator of G(f), since $E[G_{emp}(f, D_l)] = G(f)$. **Empirical risk minimization** (Vapnik, 1982; Vapnik,

1995) simply chooses

$$f = F(D_l) = \text{argmin}_{f \in \mathcal{F}} G_{emp}(f, D_l)$$

where $F(D_l)$ is the functional that maps a data set into a decision function.

An empirical estimate of $G(F(D))$, the generalization error of a functional F, can be obtained by partitioning the data in two subsets: a *training subset* D_1 to pick $f = F(D_1) \in \mathcal{F}$ which minimizes the empirical error in D_1, and a *held-out or test subset* D_2 which gives an unbiased estimate of $G(F(D_1))$. The latter is a slightly pessimistic estimate of $G(F(D))$, the generalization error associated to a functional F when applied to $D = D_1 \cup D_2$, and may be poor for small data sets. When there is not much data, it is preferable but computationally more expensive to use the K-fold cross-validation procedure (Bishop, 1995; Efron and Tibshirani, 1993).

However, in the case where the data are not i.i.d., the results of learning theory are not directly applicable, nor are the procedures for estimating generalization error.

Consider a sequence of points z_1, z_2, ..., with $z_t \in \mathcal{R}^{n+1}$, generated by an unknown process such that the z_t's may be dependent and have different distributions. Nevertheless, at each time step t, in order to make a prediction, we are allowed to choose a function f_t from a set of functions $\mathcal{F}$ using the past observations $z_1^t = (z_1, z_2, \ldots, z_t)$, i.e., we choose $f_t = F(z_1^t)$. In our applications z_t is a pair (x_t, y_t) and the functions $f \in \mathcal{F}$ take an x as input to take a decision that will be evaluated against a y through the loss function $Q(f, z)$, with $z = (x, y)$. In this paper, we consider the quadratic loss

$$Q(f, Z_t) = Q(f, (X_t, Y_t)) = (Y_t - f(X_t))^2.$$

We then define the expected generalization error G_t for the decision at time t as

$$G_t(f) = E[Q(f, Z_{t+h})|Z_1^t] = \int Q(f, z_{t+h}) P_{t+h}(z_{t+h}|Z_1^t) dz_{t+h}. \tag{4.3}$$

Here we call h the **horizon** because it corresponds to the prediction horizon in the case of prediction problems. More generally it is the number of time steps from a

decision to the time when the quality of this decision can be evaluated. The objective of learning is to find, on the basis of empirical data z_1^t, the function $f \in \mathcal{F}$ which has the lowest expected generalization error $G_t(f)$.

The process Z_t may be non-stationary, but as long as the generalization errors made by a good model are rather stable in time, we believe that one can use the data z_1^t to pick a function which has worked well in the past and hope it will work well in the future.

We will extend the above empirical and generalization error (equations 4.2 and 4.1). However we consider not the error of a single function f but the error associated with a functional F which maps a data set $D_t = z_1^t$ into a function $f \in \mathcal{F}$.

Now let us first consider the empirical error which is the analogue for non *i.i.d.* data of the K-fold cross-validation procedure. We call it the **sequential cross-validation** procedure and it measures the out-of-sample error of the functional F as follows:

$$C_T(F, z_1^T, h, M) = C_T(F, z_1^T) = \frac{1}{T-M-h+1} \sum_{t=M}^{T-h} Q(F(z_1^t), z_{t+h}) \tag{4.4}$$

where $f_t = F(z_1^t)$ is the choice of the training algorithm using data z_1^t (see equation 4.7 below), and $M > 0$ is the minimum number of training examples required for $F(z_1^M)$ to provide meaningful results.

We define the generalization error associated to a functional F for decisions or predictions with a horizon h as follows:

$$\begin{aligned} E_{Gen}(F) &= E[C_T(F, z_1^T)] = \int \frac{1}{T-M-h+1} \sum_{t=M}^{T-h} Q(F(z_1^t), z_{t+h}) P(z_1^T) dz_1^T \\ &= \frac{1}{T-M-h+1} \sum_{t=M}^{T-h} E[G_t(F(Z_1^t))] \end{aligned} \tag{4.5}$$

where $P(z_1^T)$ is the probability of the sequence Z_1^T under the generating process. In that case, we readily see that (4.4) is the empirical version of (4.5), that is (4.4) estimates (4.5) by definition. In the case of the quadratic loss, we have

$$E_{Gen}(F) = \frac{\sum_{t=M}^{T-h} E\left[Var[F(Z_1^t)(X_{t+h}) - Y_{t+h} | X_1^T] + E^2[F(Z_1^t)(X_{t+h}) - Y_{t+h} | X_1^T]\right]}{T-M-h+1} \tag{4.6}$$

To complete the picture, let us simply mention that the functional F may be chosen as

$$F(z_1^t) = \text{argmin}_{f \in \mathcal{F}} R(f) + \sum_{s=1}^{t} Q(f, z_s) \tag{4.7}$$

where $R(f)$ might be used as a regularizer, to define a preference among the functions of $\mathcal{F}$, e.g., those that are smoother.

For example, consider a sequence of observations $z_t = (x_t, y_t)$. A simple class of functions $\mathcal{F}$ is the class of "constant" functions, which do not depend on the argument x, i.e., $f(x) = \mu$. Applying the principle of empirical risk minimization to this class of function with the quadratic loss $Q(f, (x_t, y_t)) = (y_t - f(x_t))^2$ yields

$$f_t^{const} = F^{const}(z_1^t) = \text{argmin}_{\mu} \sum_{s=1}^{t} (y_s - \mu)^2 = \bar{y}_t = \frac{1}{t} \sum_{s=1}^{t} y_s, \tag{4.8}$$

the historical average of the y's up to the current time t. We call this "unconditional" predictor the *naive model*, and its average out-of-sample error is $C_T(F^{const}, z_1^T) = \frac{1}{T-M-h+1} \sum_{t=M}^{T-h} (\bar{y}_t - y_{t+h})^2$.

4.4 *Comparison of generalization abilities*

To compare the generalization ability of two functionals F_1 and F_2, let us introduce two measures of performance [4]

$$D_o = D_o(F_1, F_2) = E_{Gen}(F_2) - E_{Gen}(F_1) = E[C_T(F_2, z_1^T)] - E[C_T(F_1, z_1^T)], \tag{4.9}$$

$$R_o = R_o(F_1, F_2) = 1 - \frac{E_{Gen}(F_1)}{E_{Gen}(F_2)} = 1 - \frac{E[C_T(F_1, z_1^T)]}{E[C_T(F_2, z_1^T)]} = \frac{D_o(F_1, F_2)}{E_{Gen}(F_2)}, \tag{4.10}$$

where $E_{Gen}(.)$, $C_T(.,.)$ were discussed in the previous section. Typically, we will consider cases where $F_2 \subset F_1$. For example, $F_2 = F^{const}$ could serve as benchmark to a more complexe functional F_1. R_o and D_o will be negative, null or positive according to whether the functional F_1 generalizes worse, as well or better than F_2. Related

[4] Arguments of R_o and D_o will often be omitted to ease notation.

definitions of measure of forecast accuracy have been proposed by many authors. See (Diebold and Lopez, 1996) for a review and (Diebold and Kilian, 1997) for a general discussion. Note that, unlike D_o, R_o is unitless and therefore easier to interpret.

Broadly speaking, for an arbitrary F, **when $R_o(F, F^{const})$ or $D_o(F, F^{const})$ is positive it means that there is a dependency between the inputs and the outputs**. In other words, when there is no dependency and we use a model (F) with more capacity (e.g., degrees of freedom, $F \supset F^{const}$) than the naive model, then R_o **will be negative**. The converse is not true, i.e. $R_o < 0$ does not imply no dependency but suggests that the dependency (signal) is small relative to overall random variation (noise). So in cases where the "signal-to-noise-ratio" is small, it may be preferable not to try to capture the signal to make predictions.

The empirical versions or estimators of R_o and D_o are the statistics

$$\hat{D}_o = \hat{D}_o(F_1, F_2) = C_T(F_2, z_1^T) - C_T(F_1, z_1^T) = \frac{\sum_{t=M}^{T-h}(e_t^{F_2})^2 - \sum_{t=M}^{T-h}(e_t^{F_1})^2}{T - M - h + 1} \quad (4.11)$$

and

$$\hat{R}_o = \hat{R}_o(F_1, F_2) = 1 - \frac{C_T(F_1, z_1^T)}{C_T(F_2, z_1^T)} = 1 - \frac{\sum_{t=M}^{T-h}(e_t^{F_1})^2}{\sum_{t=M}^{T-h}(e_t^{F_2})^2}, = \frac{\hat{D}_o(F_1)}{C_T(F_2, z_1^T)} \quad (4.12)$$

where

$$e_t^F = y_{t+h} - F(z_1^t)(x_{t+h})$$

denotes the prediction error made on y_{t+h} by the functional F. This empirical $\hat{R}_o$ ($\hat{D}_o$) is a "noisy" estimate (due to the finite sample), and thus might be positive even when R_o (D_o) is negative (or vice-versa). While $E[\hat{D}_o] = D_o$, $E[\hat{R}_o] \neq R_o$ because the expection of a ratio is not equal to the ratio of expectations. In fact we should expect $\hat{R}_o$ to underestimate R_o. This means that $\hat{R}_o$ tends to be a conservative estimate of R_o, which is not undesirable. It is therefore important to analyze how "noisy" this estimate is in order to conclude on the dependency between the inputs and the outputs. This matter will be addressed in a later section.

An example may clarify all of the above. Take $n = 1$ and let $\mathcal{F}^{lin}$ be the set of affine functions, i.e. linear models $f(x) = \alpha + \beta x$. Sticking with the quadratic loss with no regularization, we have that

$$f_t^{lin}(x) = F^{lin}(z_1^t)(x) = \hat{\alpha}_t + \hat{\beta}_t x,$$

where $(\hat{\alpha}_t, \hat{\beta}_t)$, minimizing

$$\sum_{s=1}^{t}(y_s - \alpha - \beta x_s)^2,$$

are the least square estimates of the linear regression of y_s on x_s, $s = 1, \ldots, t$, and rely only on data known up to time t, i.e. z_1^t. We thus have

$$e_t^{F^{const}} = y_{t+h} - F^{const}(z_1^t)(x_{t+h}) = y_{t+h} - \bar{y}_t,$$

$$e_t^{F^{lin}} = y_{t+h} - F^{lin}(z_1^t)(x_{t+h}) = y_{t+h} - \hat{\alpha}_t - \hat{\beta}_t x_{t+h}.$$

If we assume that the Z_t's are independent with expectation $E[Y_t|x_t] = \alpha + \beta x_t$ and variance $Var[Y_t|x_t] = \sigma^2$, then (4.6) yields

$$(T - M - h + 1)E_{Gen}(F^{const}) = \sigma^2 \sum_{t=M}^{T-h}\left[1 + \frac{1}{t}\right] + \beta^2 \sum_{t=M}^{T-h} E[(X_{t+h} - \bar{X}_t)^2]$$

and

$$(T - M - h + 1)E_{Gen}(F^{lin}) = \sigma^2 \sum_{t=M}^{T-h}\left[1 + \frac{1}{t}\right] + \sigma^2 \sum_{t=M}^{T-h} E\left[\frac{(X_{t+h} - \bar{X}_t)^2}{\sum_{s=1}^{t}(X_s - \bar{X}_t)^2}\right],$$

where $\bar{X}_t = t^{-1}\sum_{s=1}^{t} X_s$ is the mean of the X's up to X_t. We then see that $R_o(F^{lin}, F^{const})$ is negative, null or positive according to whether $\frac{\beta^2}{\sigma^2}$ is smaller, equal or greater than

$$\theta = \frac{\sum_{t=M}^{T-h} E\left[\frac{(X_{t+h} - \bar{X}_t)^2}{\sum_{s=1}^{t}(X_s - \bar{X}_t)^2}\right]}{\sum_{t=M}^{T-h} E[(X_{t+h} - \bar{X}_t)^2]}. \tag{4.13}$$

This illustrates the comment made earlier regarding the fact that $R_o < 0$ means that the "signal-to-noise-ratio" ($\frac{\beta^2}{\sigma^2}$ here) is too small for F^{lin} to outperform F^{const}. Thus

if the true generating model has $\frac{\beta^2}{\sigma^2} < \theta$, a model trained from a class of models with $\beta = 0$ (the naive model) should be chosen for its better generalization, rather than a model from a class of models that allows $\beta \neq 0$. It also illustrates the point made in the introduction that when the amount of data is finite, choosing a model according to its expected generalization error may yield a different answer than choosing a model that is closest to the true generating model. See also (Vapnik, 1982) (section 8.6) for an example of the difference in out-of-sample generalization performance between the model obtained when looking for the true generating model versus choosing the model which has a better chance to generalize (in this case using bounds on generalization error, for polynomial regression).

Let us now consider a more complex case where the distribution is closer to the kind of data studied in this paper. If we assume that $E[Y_t|x_1^T] = \alpha + \beta x_t$ and $Var[Y_t|x_1^T] = \sigma^2$ with $Cov[Y_t, Y_{t+k}|x_1^T] = 0$ whenever $|k| \geq h$, then (4.6) yields

$$(T - M - h + 1)E_{Gen}(F^{const}) = \sum_{t=M}^{T-h}(\sigma^2 + E[Var[\bar{Y}_t|X_1^T]]) + \beta^2 \sum_{t=M}^{T-h} E[(X_{t+h} - \bar{X}_t)^2]$$

and

$$(T - M - h + 1)E_{Gen}(F^{lin}) = \sum_{t=M}^{T-h}(\sigma^2 + E[Var[\bar{Y}_t + \hat{\beta}_t(X_{t+h} - \bar{X}_t)|X_1^T]]).$$

We then see that R_o is negative, null or positive according to whether $\frac{\beta^2}{\sigma^2}$ is smaller , equal or greater than

$$\theta = \frac{\sigma^{-2}\sum_{t=M}^{T-h} E\left[Var[\bar{Y}_t + \hat{\beta}_t(X_{t+h} - \bar{X}_t)|X_1^T] - Var[\bar{Y}_t|X_1^T]\right]}{\sum_{t=M}^{T-h} E[(X_{t+h} - \bar{X}_t)^2]}. \tag{4.14}$$

Note that it can be shown that the above numerator is free of σ as it involves only expectations of expressions in X_t's (like the denominator).

4.5 *The financial data and preliminary results*

Experiments on the out-of-sample statistics and related in-sample statistics where performed on a financial time-series. The data is based on the daily total return,

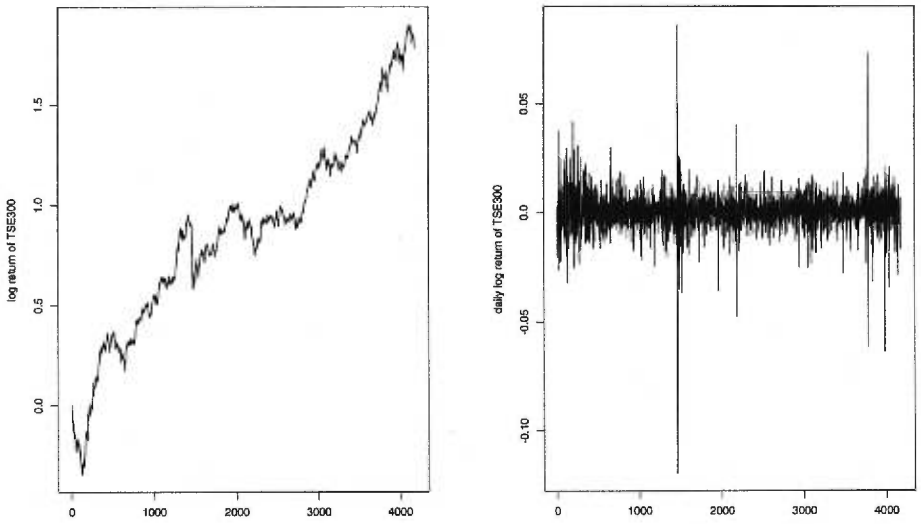

Figure 4.1.

Left: daily logarithm of TSE300 index from January 1982 to end of July 1998.
Right: daily log returns of TSE300 for the same period

including capital gain as well as dividends, for the Toronto Stock Exchange TSE300 index, starting in January 1982 up to July 1998. The total return series $TR_t, t = 0, 1, \ldots, 4178$, can be described as the result at day t of an initial investment of 1 dollar and the reinvestment of all dividends received.

We construct, for different values of h, the log-return series on a horizon h

$$r_t(h) = \log\left(\frac{TR_t}{TR_{t-h}}\right) = \log(TR_t) - log(TR_{t-h}). \tag{4.15}$$

Thus $r_t(h)$ represents the logarithm of the total return at day t of the past h day(s).

There are 4179 trading days in the sample. We consider that there are twenty-one trading days per "month" or 252 trading days per year. The real number of trading days, where the trading activities can occur, can vary slightly from month to month, depending on holidays or exceptional events, but 21 is a good approximation if we want to work with a fixed number of trading days per month. A horizon of H months

will mean $h = H \times 21$ days.

Using and predicting returns on a horizon greater than the sampling period creates an overlapping effect. Indeed, upon defining the **daily log-returns**

$$r_t = r_t(1), t = 1, \ldots, 4178,$$

we can write

$$\begin{aligned} r_t(h) &= \log(TR_t) - \log(TR_{t-h}) = \sum_{s=t-h+1}^{t} (\log(TR_s) - \log(TR_{s-1})) \\ &= \sum_{s=t-h+1}^{t} r_s \end{aligned} \tag{4.16}$$

as a moving sum of the r_t's.

We will work on monthly returns as it has been suggested from empirical evidence (Campbell et al., 1997; Fama and French, 1988) that they can be useful for forecasting, while such results are not documented for daily returns. So our horizons will be multiples of 21 days. Data are slightly better behaved when we take monthly returns instead of daily ones. For instance, the daily return series is far from being normally distributed. It is known that stock indices return distributions present more mass in their tails than the normal distribution (Campbell et al., 1997). But returns over longer horizons get closer to normality, thanks to equation 4.16 and the central limit theorem. For example, table 4.1 shows the sample skewness and kurtosis for the daily, monthly and quarterly returns. We readily notice that these higher moments are more in line with those of the normal distribution (skewness=0, kurtosis=3) when we consider longer term returns instead of daily returns.

Table 4.1 is the first illustration of the touchy problem of the overlapping effect. For instance, you will notice that the standard deviation are not the same for daily and monthly returns. This is because the daily returns statistics are based on $r_1, \ldots, r_{4178}$, whereas their monthly counterparts are based on $r_{21}(21), r_{42}(21), \ldots r_{21\times198}(21)$, that is approximatively 21 times fewer points than in the daily case. The reason for this

log-returns	skewness	kurtosis
daily	-1.22 (0.04)	33.17 (0.08)
monthly	-1.13 (0.17)	10.63 (0.35)
quarterly	-0.40 (0.30)	3.93 (0.60)

Table 4.1.

Sample skewness and sample kurtosis of TSE300 daily, monthly and quarterly log-returns. The statistics and their standard deviations (shown in parenthesis) have been computed according to formulas described in (Campbell et al., 1997).

is that we want independent monthly returns. If we assumed that the daily returns were independent, then monthly returns would have to be at least one month apart to be also independent. For instance, $r_{21}(21)$ and $r_{40}(21)$ would not be independent as they share r_{20} and r_{21}. Therefore, if we want to access independence of successive monthly returns, we have to compute the correlation coefficient between $r_{t+21}(21)$ and $r_t(21)$, or between $r_{t+h}(h)$ and $r_t(h)$ for more general h's.

Figure 4.2 left shows the square of the correlation coefficient obtained on the TSE data for $H = 1, 2, \ldots, 24$. Figure 4.2 right depicts the values of $\hat{R}_o$ with $z_t = (r_{t+h-1}(h), r_{t+2h-1}(h))$ obtained on the same data. It measures the ability of the past H month return to forecast the future H month return. According to the first plot there appears to be little relationship between past and future returns except, perhaps, when we aggregate the returns on a period of about one year ($H = 12$). Figure 4.2 right tells a similar story: at best, predictability of future returns seems possible only for yearly returns or so. But how can we decide (formally) if there is a relationship between past and future returns, and if such a relationship might be useful for forecasting? This will be the goal of the next section.

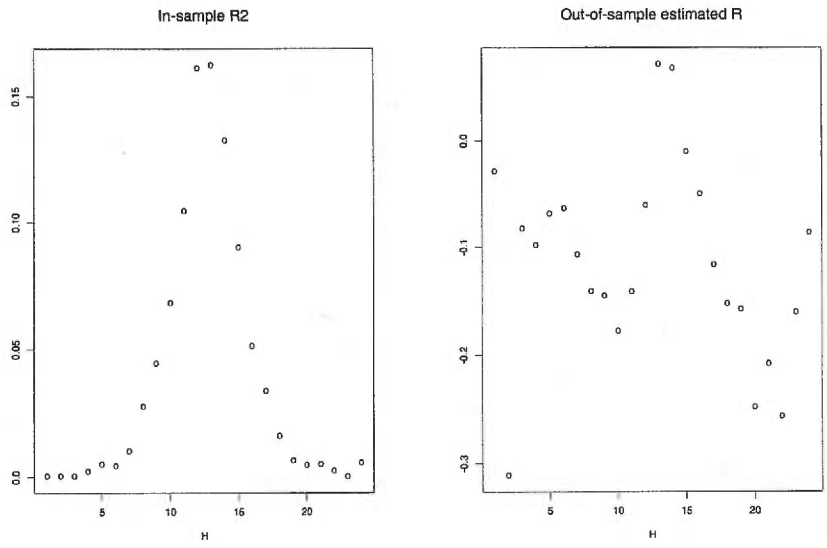

Figure 4.2.

Left: The evolution of squared correlation with the aggregation period suggests that the stronger input/output relation is for the aggregation period of around a year.

Right: Evolution of $\hat{R}_o$ with the aggregation period.

4.6 Testing the hypothesis of no relation between Y and X

Consider testing the hypothesis that there is no relationship between successive returns of horizon h, i.e. $H_0 : E[r_t(h)|r_{t-h}(h)] = \mu$. Note that $r_t(h)$ and $r_{t-h}(h)$ do not overlap but are contiguous h day returns. To put it in section 4.4's notation, we have $x_t = r_{t+h-1}(h)$ and $y_t = r_{t+2h-1}(h)$, so that, for instance, $x_1 = r_h(h)$ is the first observable x. We wish to test $E[Y_t|x_t] = \mu$.

As mentioned in the introduction, this hypothesis is not what we are actually interested in, but what we do in this section proves to be useful in section 4.7 as it allows us to introduce the bootstrap, among other things.

To perform a test of hypothesis, one needs a statistic with a behavior that depends on whether H_0 is true or false. We will mainly consider two statistics here. First we have R_o that will take smaller values under H_0 than otherwise. The other approach

to testing H_0 is to notice that if $E[r_t(h)|r_{t-h}(h)]$ does not depend on $r_{t-h}(h)$ then the correlation between $r_{t-h}(h)$ and $r_t(h)$ is null, $\rho(r_t(h), r_{t-h}(h)) = 0$. Thus we will use $\hat{\rho}(r_t(h), r_{t-h}(h))$, an estimator of $\rho(r_t(h), r_{t-h}(h))$, to test H_0 as it will tend to be closer to 0 under H_0 than otherwise.

The second thing needed in a test of hypothesis is the distribution of the chosen statistic under H_0. This may be obtained from theoretical results or approximated from a bootstrap as explained later. In the case of $\hat{\rho}(r_t(h), r_{t-h}(h))$, we do have such a theoretical result (Bartlett, 1946; Anderson, 1984; Box and Jenkins, 1970). First let us formally define

$$\hat{\rho}(r_t(h), r_{t-h}(h)) = \frac{\sum_{2h}^{T}(r_t(h) - \bar{r}(h))(r_{t-h}(h) - \bar{r}(h))}{\sum_{h}^{T}(r_t(h) - \bar{r}(h))^2}, \tag{4.17}$$

with $\bar{r}(h)$ being the sample mean of $r_h(h), \ldots, r_T(h)$. Assuming that the r_t's are independent and identically distributed with finite variance then

$$\sqrt{T-h+1}(\hat{\rho}(r_t(h), r_{t-h}(h)) - \rho(r_t(h), r_{t-h}(h))) \longrightarrow N(0, W)$$

with

$$W = \sum_{v=1}^{\infty}(\rho_{v+h} + \rho_{v-h} - 2\rho_h\rho_v)^2, \tag{4.18}$$

where ρ_k stands for $\rho(r_{t+k}(h), r_t(h))$. If the r_t's are uncorrelated with constant variance and the $r_t(h)$ are running sums of r_t's as shown in equation 4.16, then

$$\rho_k = \frac{(h - |k|)_+}{h} \tag{4.19}$$

where $u_+ = \max(u, 0)$. Therefore we have

$$W = \sum_{v=1}^{2h-1}\rho_{v-h}^2 = \sum_{v=1-h}^{h-1}\rho_v^2 = 1 + 2h^{-2}\sum_{v=1}^{h-1}(h-v)^2 = 1 + \frac{(h-1)(2h-1)}{3h}$$

where the identity $1^2+2^2+3^2+\ldots+N^2 = \frac{N(N+1)(2N+1)}{6}$ was used in the last equality. Large values of $\sqrt{\frac{T-h+1}{W}}|\hat{\rho}(r_t(h), r_{t-h}(h))|$ are unfavorable to H_0 and their significance are obtained from a $N(0,1)$ table.

The distributions of our out-of-sample statistics are unknown. However we may find an approximation by simulation (bootstrap). So we have to generate data from the hypothesis $H_0 : E[Y_t|x_t] = \mu$ (i.e. do not depend on x_t). This can be done in at least four ways.

1. Generate a set of independent r_t's and compute the $Y_t = r_{t+2h+1}(h)$'s and the $x_t = r_{t+h+1}(h)$'s in the usual way.

2. Keep the Y_t obtained from the actual data, but compute the x_t's as suggested in 1.

3. Keep the x_t's obtained from the actual data, but compute the Y_t as suggested in 1.

4. Generate a set of independent r_t's and compute the Y_t's. Then generate another set of r_t's independently of the first set and compute the x_t's.

The generation of the r_t's may come from the empirical distribution of the actual r_t's (i.e. re-sampling with replacement) or another distribution deemed appropriate. We have considered both the empirical distribution and the $N(0,1)$ distribution [5].

We believe that the generation scheme 1 is the most appropriate here since it looks more like the way the original data was treated: Y_t and x_t obtained from a single set of r_t's.

Once we have chosen a simulation scheme, we may obtain as many (B, say) samples as we want and thus get B independent realizations of the statistic $\hat{R}_o$. We then check if the out-of-sample statistic will take values that are large even in this case, compared to the value observed on the original data series. Formally, compute p-value= $\frac{A}{B}$ where A is the number of simulated $\hat{R}_o$ greater or equal to the

[5] Since $\hat{R}_o$, just like the usual in-sample R^2, is location-scale invariant, we don't have to bother about matching the mean and variance of the actual series.

$\hat{R}_o$ computed on the actual data. This measures the plausibility of H_0; small values of p-value indicate that H_0 is not plausible in the light of the actual data observed. Another way to use the bootstrap values of $\hat{R}_o$ is to assume that the distribution of $\hat{R}_o$ under H_0 is $N(\hat{E}[\hat{R}_o], \hat{V}[\hat{R}_o])$ where $\hat{E}[\hat{R}_o]$ and $\hat{V}[\hat{R}_o]$ are the sample mean and the sample variance of the B bootstrap values of $\hat{R}_o$. Comparing the actual $\hat{R}_o$ to this distribution yields the normalized bootstrap p-value.

For the scheme 1 method we simply compute the p-value of the observed $\hat{R}_o$ under the null hypothesis of no relationship between the inputs and the outputs, using the empirical histogram of this statistic over the bootstrap replications. When the p-value is very small, a more meaningful quantity might be the mean and the standard deviation of the statistic over the bootstrap replications to provide a z-statistic.

Of course, this bootstrap approach may be used even in the case where the (asymptotic) distribution of a statistic is known. Therefore, we will compute bootstrap p-values for the statistic $\hat{\rho}(r_t(h), r_{t-h}(h))$ as well as its theoretical p-value for comparison purposes.

4.6.1 Results on artificial data

In order to study different properties of in-sample and out-of-sample statistics, we have generated artificial data and tested the null hypothesis of no relationship on them. In this way, we can compare the power of the statistics on the same data set, where the hypothesis behind the use of the autocorrelation statistic is verified.

We chose an autoregressive process of order 1,

$$y_t = \beta y_{t-1} + \epsilon_t$$

for which we vary the coefficient β of auto-regression from a range of values between 0 and 0.1 and where ϵ_t is drawn from a normal distribution $N(0, \frac{1}{5})$. We conduct the tests on the null hypothesis for series of lengths in the set $\mathcal{T} = \{250, 500, 1000, 2000, 4000, 8000\}$.

N	ρ	R_o	D_o
250	[-0.106,0.101]	$(-\infty, 0.0012]$	$(-\infty, 0.017]$
500	[-0.075,0.072]	$(-\infty, 0.0009]$	$(-\infty,-0.016]$
1000	[-0.052,0.050]	$(-\infty,-0.0011]$	$(-\infty,-0.043]$
2000	[-0.038,0.035]	$(-\infty,-0.0009]$	$(-\infty,-0.069]$
4000	[-0.026,0.025]	$(-\infty,-0.0006]$	$(-\infty,-0.096]$
8000	[-0.019,0.018]	$(-\infty,-0.0004]$	$(-\infty,-0.119]$

Table 4.2.

Empirical critical points of three statistics estimated on series of different length.

We first generated, for each value of T in $\mathcal{T}$, five thousand series for which $\beta = 0$. For each of these series we construct the empirical distribution of 3 statistics, namely the autocorrelation $\hat{\rho}$ (equation 4.17), the out-of-sample $\hat{R}_o$ and $\hat{D}_o$.

From these empirical distributions, we estimated the "acceptance" region at significance level 10%, say $[L_{5\%}, H_{5\%}]$ for $\hat{\rho}$ and $(-\infty, H_{10\%}]$ for the out-of-sample statistics $\hat{D}_o$ and $\hat{R}_o$. For the out-of-sample statistics, we chose $M = 50$ for the minimum number of training examples (see equation 4.4). The values of these critical points are presented in table 4.2.

Having established the critical points at 10%, we now want to study the power of these tests, i.e. how each statistic is useful to reject the null hypothesis when the null hypothesis is false. For this goal, we generated two thousand series for different value of β, ranging from 0 to 0.1. We estimated on these series the value of the three statistics considered in table 4.2, and computed for the different values of β the number of times each of these statistics are outside the interval delimited by the critical values. The results are presented in table 4.3. Note that proportions in table 4.3 have standard deviations of at most $\frac{1}{2\sqrt{2000}} = 1.1\%$. We can observe from this table that the out-of-sample statistics $\hat{R}_o$ and $\hat{D}_o$ are less powerful than in-sample

statistics for the test of $H_o : \beta = 0$. For $\beta = 0$, powers are around 10% as they should be. It would appear from these results that when we want to test against the null hypothesis of no dependency, the classical in-sample tests provide more power. But we must underline again that this is not the null hypothesis of interest here.

4.6.2 Discussion of the results on financial data

In all cases $B = 1000$ bootstrap replications were generated and the out-of-sample statistic was computed on each of them with $M = 50$, yielding distributions of $\hat{R}_o$ for the null hypothesis of no relationship between input and output.

For $\hat{\rho}(r_t(h), r_{t-h}(h))$, the pure bootstrap p-values and normalized bootstrap p-values agree well, as shown in table 4.4, suggesting that the distribution of $\hat{\rho}$ is approximatively normal. However, we note a discrepancy between the theoretical p-values and the bootstrap p-values, suggesting that the asymptotic mean and/or variance is not a proper approximation for small samples. In fact, the mean of the $\hat{\rho}$ are supposed to be 0, which is not the case for finite sample. When using the value of mean obtained by bootstrap to the p-value obtained by the three method are much closer. Regarding $\hat{R}_0$, we see (table 4.4) that a similar pattern is observed for the positive $\hat{R}_0$. The pure bootstrap p-values seem to indicate a possible dependence of the near one year return on the past year return. Also, in this case, the empirical distributions of the $\hat{R}_0$ are not normal, the observed skewness on these distribution are systematically negative with values around -4, hence the normalized p-values should not be trusted. The theoretical p-values for the out-of-sample statistics are not known.

The table 4.4 also presents the results of the test conducted on the null hypothesis *no relationship between inputs and outputs* using the statistic D_o. This test statistics **rejects even more strongly the null hypothesis of no linear dependency** than the test based on $\hat{R}_o$.

T	$\hat{s}$	β					
		0	0.02	0.04	0.06	0.08	0.1
250	ρ	0.10	0.14	0.17	0.24	0.34	0.42
250	R_o	0.10	0.14	0.16	0.23	0.32	0.39
250	D_o	0.10	0.14	0.16	0.23	0.32	0.39
500	ρ	0.09	0.15	0.22	0.38	0.54	0.73
500	R_o	0.11	0.14	0.21	0.36	0.51	0.68
500	D_o	0.11	0.14	0.21	0.36	0.51	0.69
1000	ρ	0.11	0.17	0.36	0.61	0.82	0.94
1000	R_o	0.11	0.16	0.32	0.57	0.77	0.90
1000	D_o	0.11	0.16	0.33	0.57	0.77	0.90
2000	ρ	0.10	0.25	0.58	0.87	0.98	1.00
2000	R_o	0.11	0.22	0.54	0.82	0.96	0.99
2000	D_o	0.11	0.23	0.53	0.82	0.96	0.99
4000	ρ	0.11	0.37	0.81	0.98	1.00	1.00
4000	R_o	0.11	0.36	0.78	0.97	1.00	1.00
4000	D_o	0.11	0.36	0.78	0.98	1.00	1.00
8000	ρ	0.10	0.53	0.98	1.00	1.00	1.00
8000	R_o	0.10	0.49	0.96	1.00	1.00	1.00
8000	D_o	0.10	0.49	0.96	1.00	1.00	1.00

Table 4.3.

Power of 3 statistics for the hypothesis $H_0 : \beta = 0$ as a function of T and β.

H	$\hat{R}_o$	pbpv	nbpv	$\hat{D}_o$	pbpv	nbpv	$\hat{\rho}$	tpv	pbpv	nbpv
1	-0.03	0.83	0.67	-0.23	0.95	0.93	0.02	0.74	0.60	0.60
2	-0.31	0.99	0.99	-5.10	0.99	1.00	0.02	0.81	0.67	0.67
3	-0.08	0.68	0.51	-1.86	0.84	0.69	-0.02	0.84	0.95	0.98
4	-0.10	0.64	0.46	-3.19	0.83	0.66	-0.05	0.65	0.87	0.87
5	-0.07	0.30	0.31	-2.97	0.62	0.46	-0.07	0.60	0.72	0.72
6	-0.06	0.24	0.29	-3.37	0.52	0.41	-0.06	0.68	0.87	0.89
7	-0.11	0.42	0.39	-6.31	0.70	0.51	-0.09	0.56	0.78	0.75
8	-0.14	0.49	0.39	-8.79	0.69	0.50	-0.15	0.37	0.55	0.51
9	-0.15	0.47	0.39	-8.51	0.61	0.45	-0.18	0.31	0.48	0.46
10	-0.18	0.52	0.43	-9.65	0.57	0.45	-0.22	0.24	0.38	0.38
11	-0.14	0.38	0.35	-7.63	0.39	0.35	-0.26	0.18	0.27	0.26
12	-0.06	0.15	0.25	-3.48	0.17	0.27	-0.32	0.12	0.21	0.20
13	0.07	0.02	0.14	4.57	0.01	0.22	-0.32	0.14	0.20	0.21
14	0.07	0.04	0.14	4.54	0.03	0.17	-0.28	0.21	0.31	0.31
15	-0.01	0.10	0.19	-0.70	0.10	0.23	-0.23	0.33	0.58	0.55
16	-0.05	0.13	0.24	-3.43	0.14	0.25	-0.17	0.48	0.75	0.71
17	-0.11	0.24	0.29	-8.04	0.27	0.29	-0.13	0.58	0.99	0.96
18	-0.15	0.30	0.31	-11.02	0.31	0.34	-0.09	0.73	0.82	0.86
19	-0.16	0.28	0.32	-12.15	0.30	0.32	-0.05	0.85	0.74	0.75
20	-0.25	0.44	0.39	-20.01	0.44	0.39	-0.04	0.88	0.70	0.69
21	-0.21	0.37	0.37	-17.17	0.37	0.35	-0.04	0.89	0.71	0.70
22	-0.26	0.45	0.38	-20.67	0.45	0.36	-0.02	0.94	0.54	0.55
23	-0.16	0.31	0.32	-13.26	0.33	0.32	0.02	0.92	0.37	0.38
24	-0.09	0.19	0.28	-7.41	0.22	0.28	0.08	0.79	0.28	0.28

Table 4.4.

Test of the hypothesis of no relationship between inputs and outputs. Three statistics are used, and for each, pure bootstrap (pbpv) and normalized (nbpv) p-values are computed. For tests based on $\hat{\rho}$, we also present the theoretical (tpv) p-values computed by Bartlett's formula. The test based on the D_o statistic also

4.7 *Test of* $H_0 : R_o = 0$

Here we attack the problem we are actually interested in: assessing whether generalizations based on past returns are better than the generalizations of an alternative model, here the *naive* (constant) model. We consider linear forecasts, so that we want to know if F^{lin} generalizes better than F^{naive}.

Its distribution not being known, we will have to turn to the bootstrap method and simulate values of $\hat{R}_o$ computed on samples generated under $H_0 : R_o = 0$ (which is equivalent to $D_o = 0$). Strictly speaking, we want to solve the equation $D_o(\beta) = 0$. We can proceed analytically, as we will do when the Y_t and the X_t are different, or numerically when the Y_t are autoregressive. We are using the statistic D_o because its estimator $\hat{D}_o$ is without bias.

Consider first the case where

$$E[Y_t|X_t] = \alpha + \beta X_t, \tag{4.20}$$

with the X_t is an external covariate series (generated independently from the Y_t series, while the Y_t's are generated conditional on the X_t series).

We saw earlier that this amounts to $\frac{\beta^2}{\sigma^2}$ being equal to the ratio shown in (4.14). If we let the Y_t's (given x_1^T) have the correlation structure shown in (4.19), we have

$$\begin{aligned} E[Var[\bar{Y}_t|X_1^T]] &= \frac{\sigma^2}{t^2 h} \sum_{s=1-h}^{h-1} (h - |s|)(t - |s|) \\ &= \frac{\sigma^2}{t^2 h} \left[ht + 2 \sum_{s=1}^{h-1} (h - s)(t - s) \right] \\ &= \frac{\sigma^2}{t^2 h} \left[ht + 2 \sum_{s=1}^{h-1} s(t - h + s) \right] \\ &= \frac{\sigma^2}{t^2} \left[ht - \frac{(h^2 - 1)}{3} \right] \end{aligned} \tag{4.21}$$

and

$$E[Var[\bar{Y}_t + \hat{\beta}_t(X_{t+h} - \bar{X}_t)|X_1^T]] = \sigma^2 E[c'Vc],$$

where V is a $t \times t$ matrix with $V_{ij} = \frac{(h-|i-j|)_+}{h}$, and c is a $t \times 1$ vector with

$$c_i = \frac{1}{t} + \frac{(X_{t+h} - \bar{X}_t)(X_i - \bar{X}_t)}{\sum_{j=1}^{t}(X_j - \bar{X}_t)^2}, i = 1, \ldots, t.$$

If we let L be a $(t + h - 1) \times t$ matrix with $L_{ij} = I[0 \leq i - j < h]/\sqrt{h}$, then we may write $c'Vc$ as $W'W$ where $W = Lc$. This representation is useful if we need to compute $Var[F^{lin}(Z_1^t)(X_{t+h})|X_1^T] = \sigma^2 c'Vc$ for various values of t as recursive relations may be worked out in W.

Due the location-scale invariance of the $\hat{R}_o$ mentioned earlier, σ^2 and α may be chosen as one pleases (1 and 0, say). The expectations then depend obviously on the process generating the X_t's. The simplest thing to do is to assume that $X_1^T \sim \delta_{x_1^T}$, that is X_1^T can only take the value observed. This makes the expectation easy to work out. Otherwise, these expectations can be worked out via simulations.

Once X_1^T's process, α, β, σ^2 have been chosen, we generate $Z_1^T = (X_1^T, Y_1^T)$ as follows.

1. Generate X_1^T.

2. Generate $\epsilon_1, \ldots, \epsilon_T$ so that the ϵ_t's are independent of X_1^T with $Var[\epsilon_t] = \sigma^2$ and the covariance structure shown in (4.19). This may be done by generating independent variates with variance equal to $\frac{\sigma^2}{h}$ and take their moving sums with a window of size h.

3. Put $Y_t = \alpha + \beta x_{t-h} + \epsilon_t$.

The bootstrap test of $H_0 : R_o = 0$ could be performed by generating B samples in the way explained above, yielding B bootstrap values of $\hat{R}_o$. These would be used to compute either a pure bootstrap p-value or a normalized bootstrap p-value.

Needless to say that generating data under $H_{01} : R_o = 0$ is more tedious than generating data under H_{02} : no relationship between inputs and outputs. Furthermore

the above approach relies heavily on the distributional assumptions of linearity and the given form of covariance, and we would like to devise a procedure that can be extended to non-linear relationships, for example.

To get the distribution of $\hat{R}_o$ under H_{01}, we can consider an approximation saying that the distribution of $\hat{R}_o - R_o$ is the same under H_{01} and H_{02}. We will call this hypothesis the "shifted distribution" hypothesis (note that this hypothesis can only be approximately true since the domain of $\hat{R}_o$ is $(-\infty, 1]$). This means that we are assuming that the distribution of $\hat{R}_o$ under $R_o = 0$ has the same shape as its distribution under $\beta = 0$ but is shifted to the right, since $R_o < 0$ under $H_0 : \beta = 0$. If that was the case, and we are going the test the validity of this approximation later, generating $\hat{R}_o - 0$ under H_{01} would be the same as simulating $\hat{R}_o - R_o$ under H_{02}, which we have done previously without subtracting off R_o. This R_o can be obtained either analytically or estimated from the bootstrap as

$$1 - \frac{\sum_{b=1}^{B} C_T(F^{lin}, Z_1^T(b))}{\sum_{b=1}^{B} C_T(F^{naive}, Z_1^T(b))}.$$

Note, to make the notation clear, that the bootstrap $\hat{R}_o$'s are simply $1 - \frac{C_T(F^{lin}, Z_1^T(b))}{C_T(F^{naive}, Z_1^T(b))}$, $b = 1, \ldots, B$. From these $\hat{R}_o - R_o$'s, we obtain the bootstrap p-values and the normalized bootstrap p-values as usual. Note that the bootstrap p-values for H_{01} and H_{02} are the proportion of the $\hat{R}_o$'s (generated under H_{02}) that are greater than $\hat{R}_o(observed) + R_o$ and $\hat{R}_o(observed)$ respectively. Since $R_o < 0$ under H_{02}, we see that p-value$(H_{02}) \leq$ p-value(H_{01}).

4.7.1 Discussion of the results on artificial data

We present in this section results obtained by two approach. One consist to approximate the distribution of $\hat{R}_o$ under the null $\hat{R}_o = 0$ by shifting the distribution of $\hat{R}_o$ already obtained with $\beta = 0$. The second consist to generate the distribution of $\hat{R}_o$ with and approximation of the value of β associate with $\hat{R}_o = 0$.

$T \backslash \beta$	0	0.02	0.04	0.06	0.08	0.1
250	0.03	0.05	0.07	0.11	0.18	**0.22**
500	0.02	0.04	0.08	0.16	**0.29**	**0.45**
1000	0.02	0.04	0.13	**0.28**	**0.51**	**0.74**
2000	0.01	0.04	0.20	**0.53**	**0.81**	**0.95**
4000	0.01	0.07	**0.42**	**0.86**	**0.98**	**1.00**
8000	0.01	0.15	**0.74**	**0.99**	**1.00**	**1.00**

Table 4.5.

With the "shifted distribution" hypothesis, this table show the power of the $\hat{R}_o$ statistic to reject the null hypothesis $R_o = 0$ with a 10% level, for various alternative hypotheses corresponding to different values of β. The values in bold correspond to the conditions where we know that the null hypothesis is false. Observe than we are doing an error in rejecting the null hypothesis when it is true (e.g. T=250 and β=0.6 and 0.8).

In table 4.5 we show the power of the $\hat{R}_o$ statistic to reject the null hypothesis $R_o = 0$ with a 10% level, for various values of β. The simulations are conducted on the artificial data described in section 4.6.1. Critical values are those of table 4.2 except that they must be shifted to the right by 0.0084, 0.0052, 0.0032, 0.0019, 0.0011 and 0.0006 for $T = 250, 500, 1000, 2000, 4000, 8000$ respectively. Comparing to table 4.3 of section 4.6.1 we see that, as expected, the hypothesis $R_o = 0$ is more difficult to reject than the hypothesis of *no relation between inputs and outputs.*

We also estimate graphically the value of β for which the autoregressive model generate data with $\hat{R}_o = 0$, for a given length of the series. To do this, we plot the values of $\hat{D}_o$ as a function of the values of β, for each length of the series considered previously. We found the value of β for which the autoregressive model will give a $\hat{R}_o = 0$, called "critical *beta*" (β_c), are shown in table 4.6.

T	250	500	1000	2000	4000	8000
β_c	0.093	0.073	0.058	0.043	0.033	0.025

Table 4.6.

Empirical value of β_c associate with $R_o = 0$ on artificial data.

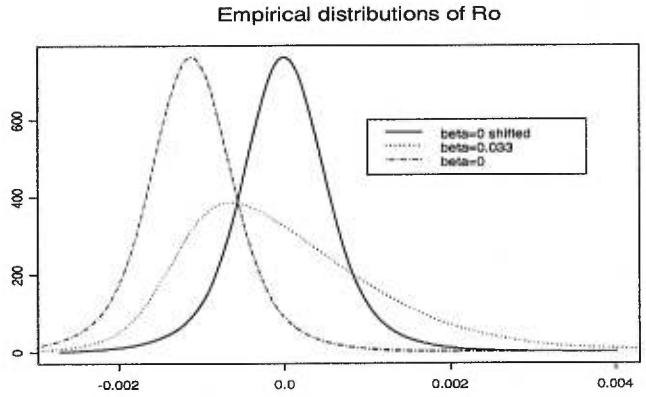

Figure 4.3. Empirical distributions of $\hat{R}_o$ for $\beta = 0$ shifted and β_c

The length of the series was 4000. *We can observe the discrepancy between the two distributions. However, note that in the right-hand tail, the shifted approximation (full line) has less mass than the true distribution (dotted line).*

After having obtained the critical value of β, we simulated, for each length in $\mathcal{T}$, 5000 series according to the autoregressive model. We compare the empirical distributions of $\hat{R}_o$ obtained here to the empirical distributions obtained by shifting the distributions of $\hat{R}_o$ obtained under $\beta = 0$. Under the approximation saying that the distribution of $\hat{R}_o - R_o$ is the same under H_{01} and H_{02}, the distributions must coincide.

We can observe from figure 4.3 that the empirical distributions of $\hat{R}_o$ obtained by shifting the distribution generated with $\beta = 0$ and the distribution of the $\hat{R}_o$ generated with $\beta = \beta_c$ do not coincide. However, we note that in the right-hand tail, the shifted approximation has less mass than the true distribution. Therefore, *if*

T	250	500	1000	2000	4000	8000
$H_{10\%}$ (shifted)	0.0096	0.0061	0.0021	0.0010	0.0005	0.0002
$H_{10\%}$ (empirical)	0.0206	0.0106	0.0058	0.0029	0.0016	0.0008

Table 4.7.

The critical points corresponding to a coverage surface of 90% for the null hypothesis $\hat{R}_o = 0$ obtained by the "shifted distribution" and by the usage of β_c obtained numerically.

we are not able to reject H_{01} using the shifted approximation, it would not have been rejected using the true distribution.

Table 4.7 also shows that using the shifted approximation the critical points for rejecting the hypothesis $\hat{R}_o = 0$ are underestimated.

In table 4.8, we see that power of the $\hat{R}_o$ statistic to reject the null hypothesis $R_o = 0$ is lower when we use the empirical distributions generated under $R_o = 0$ than when we use the shifted distributions generated under $\beta = 0$. A test based on the shifted approximation is therefore liberal (yields a lower critical value, and rejects more often than it should), but less than a test based on H_{02} (null hypothesis of no relation). This is clear by looking at the three curves in the area that is in the right-hand side tails, in Figure 4.3.

With the linear model described in equation 4.20 and using the analytical method to compute the value of β_c, we obtained the same conclusions on the critical points and the distribution of $\hat{R}_o$.

4.7.2 Discussion of the results on financial data

For different horizons, we compare the predictive ability of the linear forecast and the naive forecast, i.e. F^{lin} vs F^{naive}. The set of horizons used in the experiments was $H = 1, 3, 6, 9, \ldots, 21, 24$, in number of "months" (21 days), i.e., $h = 21H$. Ta-

$T \backslash \beta$	0	0.02	0.04	0.06	0.08	0.1
250	0.01	0.01	0.02	0.04	0.07	**0.11**
500	0.01	0.01	0.02	0.05	**0.14**	**0.24**
1000	0.00	0.01	0.04	**0.10**	**0.27**	**0.49**
2000	0.00	0.01	0.06	**0.30**	**0.61**	**0.88**
4000	0.00	0.02	**0.19**	**0.65**	**0.94**	**1.00**
8000	0.00	0.04	**0.52**	**0.97**	**1.00**	**1.00**

Table 4.8.

Power of the $\hat{R}_o$ statistic to reject the null hypothesis $R_o = 0$ with a 10% level, for various alternative hypotheses corresponding to different values of β. The values in bold correspond to the conditions where we know that the null hypothesis is false. In this case, we observe that we are rejecting the null when it is false.

ble 4.9 gives the p-value of the H_{01} hypothesis $R_o = 0$ using the method based on the "shifted distribution" hypothesis described previously. The p-values are pure histogram counts.

According to this table, there is more than 50% probability to observe values $\hat{R}_o \geq 0.07$ for horizons 13 and 14 under the null hypothesis of $R_o = 0$. Since the true critical values are likely to be larger than those computed using the shifted approximation, we conclude that **we are very far from being able to say that the inputs (past returns) are useful to make a linear forecast, even though we are able to reject the hypothesis** $\beta = 0$ (as shown previously in table 4.4).

This is a very significant result, since it shows the importance of testing a hypothesis that reflects what we really care about (e.g., out-of-sample error): testing an apparently close hypothesis (no dependency) could yield a very different conclusion!

H	pbpv	H	pbpv	H	pbpv	H	pbpv
1	0.95	7	0.85	13	0.57	19	0.81
2	0.99	8	0.87	14	0.54	20	0.84
3	0.92	9	0.86	15	0.70	21	0.83
4	0.90	10	0.89	16	0.72	22	0.83
5	0.83	11	0.86	17	0.80	23	0.80
6	0.82	12	0.76	18	0.81	24	0.76

Table 4.9.

P-values for the hypothesis $H_{01} : R_o = 0$ on the financial data. We may suppose this test to be optimistic because we use the "shifted distribution" approximation.

4.8 Conclusion

In this paper we have introduced an extension of the notion of generalization error to non-iid data. We also gave a definition of two out-of-sample statistics, R_o and D_o, to compare the forecasting ability of two models in this generalized framework, and we have presented the notion of a naive model used to establish null hypotheses.

The statistic R_o allowed us to establish a link between the signal-to-noise ratio and the particular value $R_o = 0$. We have shown that $R_o \leq 0$ means that the signal-to-noise ratio is too small for the linear functional to outperform the naive model. This does not imply no dependency but indicates that whenever the "signal-to-noise-ratio" is small, it is preferable not to try to capture the signal to make predictions.

We have made a distinction between tests for dependency and for generalization ability and we have described a method, based on bootstrap, to perform these tests.

We have used the proposed bootstrap methods to test the two null hypotheses on simulated data and on real financial data. We have used simulations to better understand the behavior of the in-sample and out-of-sample statistics in a controlled environment. We have observed that the tests based on out-of-sample statistics R_o

and D_o had less power to test against the null hypothesis of no dependency than the tests based on an in-sample statistic.

On real financial data, we have observed that we were very far from being able to say that the past returns are useful to make a linear forecast, even though we are able to reject the hypothesis of no relation. This result shows the importance of testing a hypothesis that reflects what we really care about, in that case the out-of-sample error.

In future work, we must find a way to generate the distribution of $\hat{R}_o$ under the hypothesis of no-predictability. That seems for now not trivial when the generating model is more complex than an autoregressive one or a linear model with only two parameters. But this is an important question if we want to avoid making assumptions on the distribution of the errors.

We also wish to investigate the estimation of confidence intervals for R_o. We therefore need the distribution (or at least the standard deviation) of the statistic $\hat{R}_o$ under the process generating the observed data. To do so, we would generate different series of data that preserve the dependency between the inputs and the outputs of the regression. We would use the values of $\hat{R}_o$ on these to test the null hypothesis that R_o is not positive (via confidence intervals). This method, using either parametric models or adequate non-parametric are more difficult to apply (because of the requirement that the proper input/output dependency must be preserved). Models of price returns, based on ARMA(1,1) or equivalent forms proposed in literature were studied, but it seemed in our experimentations that they did not reflect some of the statistical properties that we observed in the real price return data.

Acknowledgments

We would like to thank John W. Galbraith for his useful comments, and the NSERC funding agency for support.

Chapitre 5

CONCLUSION

Cette thèse exposait des algorithmes et des concepts pour solutionner des problèmes de prises de décisions en présence de données séquentielles. Nous avons isolé certains problèmes se présentant souvent dans des applications réelles, notamment dans des applications industrielles et financières. Nous avons montré dans le chapitre 2 qu'il peut être avantageux de combler les données manquantes par des valeurs qui optimisent le critère qui correspond à la tâche à accomplir. Nous avons développé pour montrer ce point un algorithme original utilisant un réseau de neurones récurrent où les entrées pour lesquelles les valeurs manquantes sont considérées comme des unités cachées. Nous avons montré que lorsqu'il y a des dépendances entres les variables de l'espace d'entrée, il est possible d'en tirer parti pour gérer des situations impliquant des valeurs manquantes et cela sans faire un modèle génératif de la distribution des entrées. Nous avons aussi montré comment des données arrivant à différentes échelles temporelles ou encore asynchrones peuvent être traitées comme des valeurs manquantes[1].

Nous avons aussi considéré au chapitre 3 le problème de la modélisation lorsqu'il y a peu de données disponibles, ce qui n'est pas une situation rare pour bien des cas pratiques. Nous avons développé un algorithme du genre E-M pour une mixture de mixtures de gaussiennes à paramètres partagés. Nous avons montré comment on pouvait intégrer des connaissances a priori sur les données pour résoudre une tâche pour laquelle très peu d'exemples étaient disponibles.

De plus, nous avons montré, au chapitre 4, comment on pouvait construire, à

[1] Il suffit d'avoir travaillé sur des modèles de prise de décision, notamment d'investissement, qui s'alimentent à des sources variées d'informations pour en voir les applications potentielles.

partir des erreurs de généralisation à une statistique qui nous permet de tester deux hypothèses distinctes: 1) pas de relation de la forme considérée entre les entrées et les sorties et 2) pas de prévision des sorties étant donné une classe de modèles et les entrées. Nous avons aussi souligné qu'il est parfois préférable de choisir un modèle plus simple pour faire des prédictions plutôt que de choisir un modèle dans la bonne classe de fonctions. Nous montrons qu'un modèle plus simple généralise parfois mieux que le modèle choisi dans la classe de modèles contenant le vrai modèle. Les résultats, obtenus par simulations sur des données artificielles laissent croire que les tests basés sur des statistiques hors-échantillon sont par contre un peu moins puissants que ceux basés sur les statistiques utilisant tout l'échantillon.

Nous avons basé tous ces travaux sur la recherche d'un estimé non biaisé de l'erreur qu'un modèle fera sur des données ne provenant pas de l'ensemble d'apprentissage. Nous avons étendu, au chapitre 4, la méthode de validation croisée à des données séquentielles, afin de construire un estimé honnête de l'erreur de généralisation, c'est-à-dire plus près de l'erreur qui sera faite lors de l'utilisation du modèle en temps réel. Nous pensons que pour les utilisateurs de modèles, les méthodes de construction, de sélection et d'évaluation de modèles doivent présenter une certaine fiabilité en présence de petits échantillons et ne doivent pas seulement fournir des résultats vrais asymptotiquement. Par ailleurs, les méthodes usuelles d'estimation de paramètres, loin d'être confinées à des cas de laboratoire, peuvent bénéficier de l'approche préconisée dans cette thèse.

La construction de statistiques basées sur des erreurs de généralisation ayant un analogue en statistique traditionnelle (utilisant tout l'échantillon puis testant certaines hypothèses) peut s'étendre à d'autres statistiques que le R^2 présenté au chapitre 4. Nous pensons, par exemple, aux ratios de variances (Campbell et al., 1997) utilisés pour tester l'hypothèse des marchés efficaces.

Plusieurs avenues de recherches permettant de mieux caractériser les statistiques hors-échantillons se présentent, en voici quelques-unes. D'abord, nous devons étudier

plus avant les tests sur l'hypothèse nulle $H_0 : R_o^2 = 0$ en générant des Y_t à l'aide des formules récursives présentées dans la section 4.7. Ceci peut se faire en utilisant les dividendes versées par le TSE300 ou les taux d'intérêt à court terme. De tels modèles sont proposés par les économètres et les financiers (Campbell et al., 1997). Ce faisant nous aurons un modèle où les X_t sont différents des Y_t (les rendements) et nous pourrons résoudre analytiquement l'équation $D_o(\beta) = 0$ pour obtenir la valeur de β_c.

Une autre avenue de recherche que nous pouvons aussi emprunter consiste à utiliser la statistique hors-échantillon R_o pour étudier le problème du nombre effectif d'exemples qui se présente lorsque les données ne sont pas indépendantes. Cela permettrait de contribuer au débat (Kaul, 1996) autour de la question suivante: lorsque l'on fait de la prédiction sur un long horizon, est-il préférable d'utiliser des données qui se chevauchent ou d'utiliser un ensemble, souvent beaucoup plus petit, de données qui ne se chevauchent pas?

Sur un horizon d'un an, nous n'avons pas d'évidence concluante, à l'aide du test disponible présentement, contre l'hypothèse que *les rendements passés ne sont pas utiles pour faire des prédictions à l'aide d'un modèle linéaire*, malgré le rejet de l'hypothèse *pas de relation entre les rendements passés et les rendements futurs.* Nous devons explorer plus à fond les sources de désaccord entre la statistique ρ et les statistique hors-échantillons R_o et D_o.

Nous devons développer un test de l'hypothèse $R_o = 0$ dans le cas où les modèles sont plus complexes que ceux déjà énoncés.

Afin de construire un intervalle de confiance pour $\hat{R}_o$ ou $\hat{D}_o$, nous devons obtenir la distribution de la statistique sous le processus ayant généré les données. Dans le cas des rendements des indices boursiers, le modèle générateur ne nous est pas connu et les méthodes de bootstrap qui permettraient de préserver la structure de dépendance entre les entrées et les sorties sont difficiles à mettre au point. Ces deux voies méritent d'être explorées simultanément.

Enfin, après avoir approfondi les connaissances des distributions des statistiques hors-échantillons, nous allons étendre notre étude sur la prévision des rendements d'actifs financiers en considérant d'autres actifs (par exemple d'autres indices), des séries plus longues et des modèles plus complexes.

BIBLIOGRAPHY

Ahmad, S. and Tresp, V. (1993). Some solutions to the missing feature problem in vision. In Hanson, S. J., Cowan, J. D., and Giles, C. L., editors, *Advances in Neural Information Processing Systems 5*, San Mateo, CA. Morgan Kaufman Publishers.

Almeida, L. (1987). A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In Caudill, M. and Butler, C., editors, *IEEE International Conference on Neural Networks*, volume 2, pages 609–618, San Diego 1987. IEEE, New York.

Anderson, T. (1984). *An Introduction to Multivariate Statistical Analysis.* John Wiley and Sons, New York.

Baker, J. (1975). Stochastic modeling for automatic speech understanding. In Reddy, D., editor, *Speech Recognition*, pages 521–542. Academic Press, New York.

Bareiss, E. and Porter, B. (1987). Protos: An exemplar-based learning apprentice. In *Proceedings of the 4th International Workshop on Machine Learning*, pages 12–23, Irvine, CA. Morgan Kaufmann.

Bartlett, E. and Uhrig, R. (1992). Nuclear power plant status diagnostics using an artificial neural network. *Nuclear Technology*, 97.

Bartlett, M. (1946). On the theoritical specification of sampling properties of autoc orrelated time series. *J. Royal Stat. Soc. B*, 8:27–41.

Basu, A. and Bartlett, E. (1994). Detecting faults in a nuclear power plant by using dynamic node architecture artificial neural networks. *Nuclear Science and Engineering*, 116.

Baum, L. E., Petrie, T., Soules, G., and Weiss, N. (1970). A maximization technique occuring in the statistical analysis of probabilistic functions of Markov chains. *Ann. Math. Statistic.*, 41:164–171.

Baxter, J. (1996). Learning model bias. In Mozer, M., Touretzky, D., and Perrone, M., editors, *Advances in Neural Information Processing Systems*, volume 8, pages 169–175, Cambridge, MA. MIT Press.

Becker, S. and LeCun, Y. (1989). Improving the convergence of back-propagation learning with second order methods. In Touretzky, D., Hinton, G., and Sejnowski, T., editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 29–37, Pittsburg 1988. Morgan Kaufmann, San Mateo.

Bengio, Y. (1996a). *Neural Networks for Speech and Sequence Recognition.* International Thompson Computer Press, London, UK.

Bengio, Y. (1996b). Using a financial training criterion rather than a prediction criterion. Technical Report #1019, Dept. Informatique et Recherche Operationnelle, Universite de Montreal.

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

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

Bottou, L. (1991). *Une approche théorique de l'apprentissage connexioniste; applications à la reconnaissance de la parole.* PhD thesis, Université de Paris XI.

Bottou, L. and Gallinari, P. (1991). A framework for the cooperation of learning algorithms. In Lippman, R. P., Moody, R., and Touretzky, D. S., editors, *Advances in Neural Information Processing Systems 3*, pages 781–788, Denver, CO.

Box, G. and Jenkins, G. (1970). *Time Series Analysis: Forecasting and Control.* Holden-Day, San Francisco.

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

Chauvin, Y. (1990). Dynamic behavior of constrained back-propagation networks. In Touretzky, D., editor, *Advances in Neural Information Processing Systems 2*, pages 642–649, Denver, CO. Morgan Kaufmann.

Dai, H., Lina, J., Goulard, B., Thomson, J., and Scott, C. (1995). An expert diagnostic system introducing wavelets analysis and neural network. In *1995 Robotic and Knowledge Based Sytems Workshop*, St. Hubert, Canada.

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

Devroye, L. (1988). Automatic pattern recognition: A study of the probability of error. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(4):530–543.

Diebold, F. X. and Kilian, L. (1997). Measuring predictability:theory and macroeconomics applications. *NBER technical working paper*, 213.

Diebold, F. X. and Lopez, J. A. (1996). Forecast evaluation and combination. In Maddala, G. and Rao, C., editors, *Handbook of Statistics, Vol. 14*, pages 241–268. Elsevier Science.

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

Duda, R. and Hart, P. (1973). *Pattern Classification and Scene Analysis.* Wiley, New York.

Efron, B. and Tibshirani, R. J. (1993). *An introduction to the Bootstrap.* Chapman and Hall, New-York.

Fama, E. and French, K. (1988). Permanent and temporary components of stock prices. *Journal of Political Economy*, 96(2):246–273.

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

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

Granger, C. W. J. and Newbold, P. (1976). Forecasting transformed series. *J. Roy. Statist. Soc. B*, 38:189–203.

Guyon, I., Boser, B., and Vapnik, V. (1993). Automatic capacity tuning of very large vc-dimension classifiers. In Hanson, S. J., Cowan, J. D., and Giles, C. L., editors, *Advances in Neural Information Processing Systems 5*, pages 147–155, Denver, CO. Morgan Kaufmann.

Hamilton, J. (1989). A new approach to the economic analysis of non-stationary time series and the business cycle. *Econometrica*, 57(2):357–384.

Hampshire, J. B. and Kumar, B. V. K. V. (1992). Shooting craps in search of an optimal strategy for training connectionist pattern classifiers. In Moody, J., Hanson, S., and Lippmann, R., editors, *Advances in Neural Information Processing Systems*, volume 4, pages 1125–1132, Denver, CO. Morgan Kaufmann.

Han, H.-H., Jung, H.-C., Lee, Y.-R., and Jeong, S.-C. (1996). Application of neural network for PWR steam generator water level control at low power operation. In *Proceedings of the 1996 American Nuclear Society, International Topical Meeting on Nuclear Plant Instrumentation, Control and Human-Machine Interface Technologies*, volume 1, pages 49–52. American Nuclear Society, Illinois, USA.

Hertz, J., Krogh, A., and Palmer, R. (1991a). *Introduction to the Theory of Neural Computation.* Addison-Wesley.

Hertz, J., Krogh, A., and Palmer, R. G. (1991b). *Introduction to the theory of neural computation.* Santa fe Institute studies in the Sciences of Complexity. Addison Wesley.

Hines, J. (1996). A logarithmic neural network architecture for a PRA approximation. In *Proceedings of the 1996 American Nuclear Society, International Topical Meeting on Nuclear Plant Instrumentation, Control and Human-Machine Interface Technologies*, volume 1, pages 235–241. American Nuclear Society, Illinois, USA.

Hinton, G. E., Sejnowski, T. J., and Ackley, D. H. (1984). Boltzmann machines: Constraint satisfaction networks that learn. Technical Report TR-CMU-CS-84-119, Carnegie-Mellon University, Dept. of Computer Science.

Jeong, E., Furuta, K., and Kondo, S. (1996). Identification of transient in nuclear power plant using adaptive template matching with neural network. In *Proceedings of the 1996 American Nuclear Society, International Topical Meeting on Nuclear Plant Instrumentation, Control and Human-Machine Interface Technologies*, volume 1, pages 243–250. American Nuclear Society, Illinois, USA.

Kaul, G. (1996). Predictable components in stock returns. In Maddala, G. and Rao, C., editors, *Handbook of Statistics, Vol. 14*, pages 269–296. Elsevier Science.

Kozma, R., Kitamura, M., and Sato, S. (1996). Monitoring of NPP state using structural adaptation in a neural signal processing system. In *Proceedings of the 1996 American Nuclear Society, International Topical Meeting on Nuclear Plant Instrumentation, Control and Human-Machine Interface Technologies*, volume 1, pages 273–278. American Nuclear Society, Illinois, USA.

Ku, C., Lee, K., and Eawards, R. (1992). Improved nuclear reactor temperature control using diagonal recurrent neural networks. *IEEE Transactions on Nuclear Science*, 39:2292–2308.

Lang, K. J. and Hinton, G. E. (1988). The development of the time-delay neural network architecture for speech recognition. Technical Report CMU-CS-88-152, Carnegie-Mellon University.

LeCun, Y. (1986). Learning processes in an asymmetric threshold network. In Bienenstock, E., Fogelman-Soulié, F., and Weisbuch, G., editors, *Disordered Systems and Biological Organization*, pages 233–240. Springer-Verlag, Berlin, Les Houches 1985.

LeCun, Y., Kanter, I., and Solla, S. (1991). Second order properties of error surfaces: learning time, generalization. In Lippman, R. P., Moody, R., and Touretzky,

D. S., editors, *Advances in Neural Information Processing Systems 3*, pages 918–924, Denver, CO. Morgan Kaufmann.

Lin, C., Chang, S.-C., and Lin, K.-J. (1996). Simulation of the balance of plant of a nuclear power plant by neural networks. In *Proceedings of the 1996 American Nuclear Society, International Topical Meeting on Nuclear Plant Instrumentation, Control and Human-Machine Interface Technologies*, volume 1, pages 251–255. American Nuclear Society, Illinois, USA.

Lin, J., Bartal, Y., and Uhrig, R. E. (1995). Nuclear power plant transient diagnostics using artificial neural networks that allow "don't know" classifications. *Nuclear Technology*, 110:436–449.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematics, Statistics and Probability, Vol. 1*, pages 281–296.

Morgan, N. and Bourlard, H. (1990). Generalization and parameter estimation in feedforward nets: some experiments. In Touretzky, D., editor, *Advances in Neural Information Processing Systems 2*, pages 413–416, Denver, CO. Morgan Kaufmann.

Nowlan, S. (1988). Gain variation in recurrent error propagation networks. *Complex Systems*, 2:305–320.

Parlos, A., Muthusami, J., and Atiya, A. (1994). Incipient fault detection and identification in process systems using accelerated neural network learning. *Nuclear Technology*, 105:145.

Pineda, F. (1987). Generalization of back-propagation to recurrent neural networks. *Physical Review Letters*, 59:2229–2232.

Pineda, F. (1989). Recurrent back-propagation and the dynamical approach to adaptive neural computation. *Neural Computation*, 1:161–172.

Poterba, J. and Summers, L. (1988). Mean reversion in stock prices. *Journal of Financial Economics*, 22:27–59.

Rumelhart, D., Hinton, G., and Williams, R. (1986a). Learning internal representations by error propagation. In Rumelhart, D. and McClelland, J., editors, *Parallel Distributed Processing*, volume 1, chapter 8, pages 318–362. MIT Press, Cambridge.

Rumelhart, D., Hinton, G., and Williams, R. (1986b). Learning representations by back-propagating errors. *Nature*, 323:533–536.

Rumelhart, D., McClelland, J., and the PDP Research Group (1986c). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1. MIT Press, Cambridge.

Simard, P. and LeCun, Y. (1992). Reverse TDNN: An architecture for trajectory generation. In Moody, J., Hanson, S., and Lipmann, R., editors, *Advances in Neural Information Processing Systems 4*, pages 579–588, Denver, CO. Morgan Kaufmann, San Mateo.

Tresp, V., Ahmad, S., and Neuneier, R. (1994). Training neural networks with deficient data. In Cowan, J., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems 6*, pages 128–135. Morgan Kaufman Publishers, San Mateo, CA.

Uhrig, R. (1991). Potential applications of neural networks to the operation of a nuclear power plant. *Nuclear Safety*, 32(1).

Uhrig, R. (1994). Artificial neural networks in nuclear power plants. *Nuclear News*, 37(9):38.

Vapnik, V. (1982). *Estimation of Dependences Based on Empirical Data.* Springer-Verlag, Berlin.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory.* Springer, New-York.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. (1989). Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37:328–339.

White, H. (1991). An overview of representation and convergence results for multilayer feedforward networks. In Lippman, R. P., Moody, J. E., and Touretzky, D. S., editors, *Advances in Neural Information Processing systems 3*, San Mateo, USA. Morgan Kaufmann.