ACSEG'2003 proceedings - Connectionist Approaches in Economics and Management Sciences
Nantes (France), 20-21 November 2003, pp. 153-164

PRÉDICTION DE SÉRIES TEMPORELLES FINANCIÈRES PAR DOUBLE CARTE DE KOHONEN ET MODÈLES RBFNS LOCAUX

APPLICATION À LA PRÉDICTION DE L'INDICE BOURSIER DAX30

Simon Dablemont[1], Geoffroy Simon[1], Amaury Lendasse[2],
Alain Ruttiens[3], Michel Verleysen[1]

Université catholique de Louvain
[1]DICE - Place du Levant, 3
[2]CESAME - Avenue G. Lemaître, 4
B – 1348 LOUVAIN-LA-NEUVE
BELGIQUE

[3]CBC Banque

Grand-Place, 5
B – 1000 BRUXELLES
BELGIQUE

[1]Tél : +32 10 47 25 40 [1]Fax : +32 10 47 25 98

dablemont@dice.ucl.ac.be, simon@dice.ucl.ac.be, lendasse@auto.ucl.ac.be,
alain.ruttiens@cbc.be, verleysen@dice.ucl.ac.be

Résumé : Une méthode générale de prédiction de séries temporelles est présentée. Son principe est de modéliser l'évolution passée d'une série en utilisant des cartes de Kohonen. Les régions définies par ces cartes sont ensuite modélisées par des modèles locaux. Des prédictions locales sont combinées en une seule prédiction en utilisant un modèle stochastique. Cette méthode, conçue pour la prédiction de séries financières, est appliquée à la prédiction des rendements de l'index DAX30.

Mots clés : prédiction de séries temporelles, cartes de Kohonen, réseaux à fonctions radiales de base, prédiction financière

Abstract: A time series forecasting method is presented in this paper. The method models the dynamic of the past evolution of a time series using self-organizing maps. Within the regions obtained with those maps the data are locally modelled using a Radial Basis Function networks. The local predictions are then combined in a single one using a probabilistic model. The method thought it could be applied to any kind of time series has been specifically designed for financial ones and is applied here to predict the returns of the DAX30 index.

Keywords: time series forecasting, Self-Organizing Maps, Radial Basis Function networks, financial forecasting

1 INTRODUCTION

Dans le cadre de la finance, prédire le rendement futur d'un actif est bien entendu un problème de première importance pour les opérateurs. Si quelqu'un peut connaître à l'avance le rendement qu'il va prendre sur un titre, même avec une fiabilité et une précision limitée, il peut utiliser cette information pour investir à son meilleur profit. Mais dans la réalité, fournir ce genre d'information est loin d'être aussi simple.

ACSEG'2003 proceedings - Connectionist Approaches in Economics and Management Sciences
Nantes (France), 20-21 November 2003, pp. 153-164

Plusieurs techniques ont été développées pour essayer de modéliser les séries financières. Parmi ces techniques, le modèle de marche aléatoire Gaussienne (sous la forme d'un processus géométrique de Wiener) est devenu l'un des piliers de la modélisation en finance. Régulièrement utilisé, ce modèle s'est montré suffisamment robuste et général que pour pouvoir être appliqué à (presque) n'importe quelle série temporelle. Par ailleurs, en considérant le long terme et des données suffisamment espacées dans le temps, ce modèle est généralement bien vérifié. Cependant, il est quand même mis en échec principalement dans un contexte de court voire de très court terme. Ces échecs pourraient être dus au caractère hautement non-linéaire de la série, de même qu'à l'unicité du modèle proposé lorsqu'on fait l'hypothèse d'une marche aléatoire.

En effet, une étude [1] a par exemple montré que les rendements pouvaient parfois être modélisés par deux équations différentielles. Les rendements pourraient donc être modélisés par un processus caractérisé par deux dynamiques sous-jacentes distinctes, la série évoluant de l'une à l'autre de ces dynamiques au cours du temps. Le problème de cette approche est que rien ne permet, au sein de la série elle-même, de prédire ce qui va provoquer le changement d'une dynamique vers l'autre. Généralement, ce genre de phénomène est dû à une cause externe qui influence directement la série. C'est par exemple le cas de décisions stratégiques prises par un conseil d'administration et auxquelles réagit le marché.

En pratique, les opérateurs financiers savent que les marchés ont bien plus d'un seul comportement, et que ce comportement fait bien plus que d'évoluer d'une dynamique à une autre. Ce phénomène semble en contradiction avec les méthodes financières de modélisation, et en premier lieu avec l'hypothèse de marche aléatoire Gaussienne. Ces observations prouveraient empiriquement l'utilité de considérer des modèles complexes, éventuellement non-linéaires, capables de modéliser différentes dynamiques issues d'une seule série. Pour être utilisable, ce genre de modèles ne devrait, bien entendu, considérer qu'un nombre limité de dynamiques sous-jacentes et, en plus de la modélisation de chacune de ces dynamiques, modéliser le(s) phénomène(s) qui provoque(nt) le passage de l'une à l'autre. Par ailleurs, ces modèles devraient être relativement simples à développer, et adaptés en fonction des données disponibles.

Dans ce papier, une méthode de prédiction, basée sur l'analyse empirique de l'évolution passée de la série au moyen d'outils non-linéaires (réseaux de neurones artificiels) sera présentée. Comme pour d'autres méthodes de prédiction, l'idée de base est que les données à disposition, constituant l'évolution passée de la série, contiennent suffisamment d'information pour en déterminer, au moins partiellement, son évolution future. Le but de cette méthode est de montrer empiriquement que le fait de développer un modèle recouvrant plusieurs dynamiques est capable d'améliorer les prédictions d'une série financière à court terme. Une particularité de cette méthode est qu'elle permet de modéliser le caractère potentiellement multi-dynamique de la série. Son principe est de séparer la série en classes, et de construire un modèle local, non-linéaire dans ce cas-ci, de chacune de ces classes. Tous ces modèles locaux sont ensuite combinés en utilisant un modèle probabiliste développé sur base des données disponibles. Bien que cette méthode puisse être appliquée à n'importe quelle série temporelle, le souci originel de modéliser plusieurs dynamiques la rend plus particulièrement intéressante dans le cadre de la modélisation de séries financières.

Dans la section 2, le concept de régresseur de données sera introduit. La section 3 présentera brièvement les cartes de Kohonen, alors que dans la section 4 il sera question des réseaux à fonction radiale de base. La méthode de prédiction sera alors décrite dans la section 5 et ensuite appliquée à l'index DAX30. Les résultats de cette application seront présentés dans la section 6, de même que le gain qu'on peut obtenir en prenant en compte un régresseur plus large.

2 LES RÉGRESSEURS

Dans le contexte de la prédiction de séries temporelles, les données doivent généralement être manipulées avant de pouvoir utiliser un modèle de prédiction. En effet, en utilisant la dernière donnée connue d'une série temporelle scalaire, on n'a pas suffisamment d'information pour prédire la valeur suivante avec un minimum de précision.

La manipulation la plus courante est de créer des régresseurs de données, de taille fixe. Ces régresseurs contiennent quelques données passées de la série, et sont supposés contenir le maximum d'information disponible pour réaliser une prédiction aussi précise que possible. Déterminer la taille (l'ordre) ainsi que les composantes du régresseur contenant le plus d'information possible pour un ensemble spécifique de données est une question intéressante et ouverte [2], mais qui dépasse le cadre de cet article. Par la suite, nous supposerons que le régresseur utilisé est optimal ou quasi-optimal pour un ensemble donné, et ce en se basant sur des connaissances liées au comportement physique du processus ou sur des méthodes statistiques de sélection de modèles par rééchantillonage.

Considérons une série scalaire $x(t)$, $1 \leq t \leq n$, $x(t)$ étant la valeur de la variable explicative de la série temporelle au temps t. Par abus de langage, cette variable explicative est par la suite appelée série temporelle. En supposant que le régresseur optimal soit connu et soit, par exemple, d'ordre p, on peut transformer la série de départ $x(.)$ en une série de régresseurs $X(t)$ définis selon :

$$X(t) = (x(t-p+1), x(t-p+2), \ldots, x(t), u(t-q+1), u(t-q+2), \ldots, u(t)) \quad (1)$$

avec $u(t)$ la valeur, au temps t, d'une variable exogène qui possède une information utile pour obtenir la prédiction de la série. Bien entendu, la notation ci-dessus peut être étendue pour prendre en compte v variables exogènes, avec des ordres respectifs q_1, q_2, …, q_v. Le cas d'une unique variable exogène sera retenu ici, pour des raisons de simplicité des notations. Calculer les régresseurs $X(t)$ d'une série temporelle revient donc à transformer une série scalaire $x(.)$ comportant n données ainsi qu'une variable exogène en une série de n-p+1 données de dimension p+q.

3 LES CARTES DE KOHONEN

Les cartes de Kohonen (self-organizing maps ou SOM dans la littérature [3]) peuvent être définies comme un algorithme de classification non supervisé issu du domaine des réseaux de neurones artificiels. L'application de cet algorithme crée un ensemble, de taille finie et fixée a priori, de prototypes ayant les mêmes dimensions que les données en entrée. Une relation de voisinage physique relie ces prototypes, ou vecteurs codes, entre eux sur une carte uni-dimensionnelle (ficelle) ou bi-dimensionnelle (carte carrée ou rectangulaire).

Après l'apprentissage [3], chaque prototype représente un sous-ensemble de l'ensemble des entrées, partageant certaines caractéristiques. En utilisant la terminologie de Voronoï, un prototype correspond au centroïde d'une zone ou d'une région, chaque région étant l'une des classes obtenues par l'algorithme. La carte a donc comme propriété de réaliser une quantification vectorielle de l'espace des entrées tout en respectant la distribution originale de ces entrées. Une autre propriété est que les prototypes obtenus après application de l'algorithme sont ordonnés selon leur position au sein de l'espace des entrées. Par conséquent, des entrées ayant les mêmes caractéristiques sont projetées soit sur un même prototype (comme c'est le cas dans les autres méthodes de quantification vectorielle), soit sur deux prototypes voisins au niveau de la carte. Cette projection consiste à déterminer le prototype le plus proche selon une mesure de distance, généralement euclidienne. Cette dernière propriété, connue sous le nom du respect de la topologie, n'est pas vérifiée par d'autres méthodes de quantification vectorielle.

Bien que n'importe quelle méthode de classification aurait pu être utilisée, nous avons choisi d'utiliser des cartes de Kohonen parce qu'il est relativement facile de représenter graphiquement les prototypes et les classes. Intuitivement, les grilles uni- ou bi-dimensionnelle peuvent être vues comme des espaces à une ou deux dimension(s) dans lesquels les données en entrées sont projetées par les SOMs.

4 LES RÉSEAUX À FONCTION RADIALE DE BASE

Les réseaux à fonction radiale de base (Radial Basis Function Networks ou RBFN dans la littérature) sont des réseaux utilisés aussi bien en approximation de fonction qu'en classification. Tout comme les perceptrons multicouches (Multi-Layer Perceptrons ou MLP), les RBFNs possèdent la propriété théorique d'approximateur universel [4].

Un réseau RBFN est un réseau dont les unités en entrée sont connectées avec toutes les unités de la couche cachée, unique dans le cas du RBFN. Chacune de ces unités de la couche cachée effectue une transformation non-linéaire des vecteurs en entrée. La sortie du RBFN est finalement obtenue en prenant une somme pondérée des non linéarités obtenues au niveau de la couche cachée. Généralement, les non linéarités sont des fonctions Gaussiennes. L'équation d'un réseau RBFN se résume donc en :

$$\hat{y} = \sum_{i=1}^{k} \lambda_i e^{\left(-\frac{\|x - c_i\|^2}{2\sigma_i^2} \right)} \qquad (2)$$

où x est le vecteur en entrée, $\hat{y}$ est la sortie scalaire du réseau RBFN, c_i représentent les centres des fonctions Gaussiennes, σ_i leurs largeurs respectives, et λ_i les poids du réseau. Ces poids correspondent à l'importance relative de chaque noyau dans la sortie du réseau $\hat{y}$. Un des avantages des réseaux RBFN comparés à d'autres types de réseaux de neurones, c'est que les trois ensembles de paramètres c_i, σ_i, et λ_i, peuvent être appris de façon indépendante et même relativement rapide, notamment pour les poids λ_i qui peuvent être obtenus comme solution d'un système d'équations linéaires. Les détails concernant l'apprentissage d'un réseau RBFN sont décrits dans la littérature [5, 6].

5 LA MÉTHODE DE PRÉDICTION

Dans cette section, la méthode de prédiction va être décrite en détails sur le cas simple d'une série temporelle scalaire, pour des raisons de clarté de la présentation. La méthode peut être généralisée facilement au cas des séries non scalaires. Dans un premier temps, la méthode sera décrite de façon intuitive, chacune des principales étapes étant ensuite plus précisément expliquée.

5.1 Description de la méthode

Comme pour d'autres méthodes de prédiction, l'idée de base est de modéliser le passé de la série. Pour prédire une valeur en $x(n+1)$, on construit un régresseur $X(n)$ avec les données, connues, jusqu'en $x(n)$. On recherche ensuite, dans le passé de la série, le (ou les) régresseur(s) existant(s) qui lui ressemble(nt) le plus. On observe alors la (les) véritable(s) valeur(s) suivante(s) de ce (ces) régresseur(s). En combinant cette (ces) vraie(s) valeur(s) suivante(s), on obtient une estimation de la valeur cherchée $x(n+1)$, estimation notée $\hat{x}(n+1)$.

Plus précisément, on construit d'abord les régresseurs $X(t)$ d'ordre optimal p, supposé connu. Ces régresseurs sont quantifiés en utilisant une première carte de Kohonen. Ensuite, on étend les régresseurs $X(t)$ pour leur faire contenir la véritable valeur suivante dans la série $x(t+1)$. Ces régresseurs étendus, notés $X'(t)$, sont eux aussi quantifiés en utilisant une seconde carte de Kohonen. Le lien entre les deux cartes de Kohonen, modélisant la dynamique de la série, est repris dans une table de fréquence. Enfin, chacune des classes de la seconde carte de Kohonen, contenant des régresseurs étendus, est modélisée par un réseau RBFN, qui approxime localement la relation entre les $p+q$ composantes du régresseur et la dernière composante $x(t+1)$ du régresseur étendu. Finalement, pour prédire, on combine les prédictions locales obtenues par les différents modèles RBFNs de chaque classe en fonction des fréquences respectives contenues dans la table de fréquence.

5.2 Calcul des régresseur

La première étape de la méthode consiste à calculer des régresseurs $X(t)$ de la série temporelle. Ce calcul est décrit par la relation (1) et détaillé dans la section 2. Par la suite, les vecteurs $X(t)$ seront appelés les entrées de la méthode.

5.3 Quantification des régresseurs $X(t)$

L'algorithme de Kohonen est appliqué aux entrées vectorielles $X(t)$ de dimension $p+q$. Après convergence de l'algorithme, on dispose d'une carte sur les vecteurs en entrée, appelée par la suite IN, comportant N_{in} classes et autant de vecteurs codes IN_i, $1 \leq i \leq N_{in}$, associés à chacune des classes.

5.4 Calcul des régresseur étendus

Sur base des régresseurs $X(t)$ de la série temporelle, on calcule $X'(t)$, aussi appelés les sorties de la méthode, en incluant la vraie valeur suivante $x(t+1)$dans la série temporelle :

$$X'(t) = (x(t-p+1), x(t-p+2), \ldots, x(t), x(t+1), u(t-q+1), u(t-q+2), \ldots, u(t)). \qquad (3)$$

Notons que, par construction, à chaque régresseur étendu $X'(t)$, correspond un régresseur $X(t)$; $X'(t)$ et $X(t)$ possèdent $p+q$ composants en commun.

5.5 Quantification des sorties

L'algorithme de Kohonen est à nouveau appliqué aux entrées vectorielles $X'(t)$, de dimension $p+q+1$. Après convergence de l'algorithme, on dispose d'une deuxième carte de Kohonen, la carte *OUT*, comportant N_{out} classes et autant de vecteurs codes OUT_j, $1 \leq j \leq N_{out}$, associés à chacune des classes.

5.6 La table de fréquence

Les deux cartes de Kohonen *IN* et *OUT* ne contiennent qu'une information statique relative à l'évolution passée de la série. Pour modéliser le passage d'une donnée à l'autre, soit la dynamique de la série, on construit une table de fréquence $F(i, j)$, $1 \leq i \leq N_{in}$, $1 \leq j \leq N_{out}$, reprenant les liens entre chaque classe i de la carte *IN* et chaque classe j de la carte *OUT* :

$$F(i,j) = \frac{\#\{X'(t) \in OUT_j \mid X(t) \in IN_i\}}{\#\{X(t) \in IN_i\}}, \qquad (4)$$

avec $1 \leq i \leq N_{in}$, $1 \leq j \leq N_{out}$. Bien entendu, ces fréquences $F(i, j)$ sont calculées empiriquement sur base des données de la série temporelle étudiée.

Intuitivement, les éléments $F(i, j)$ de cette table représentent toutes les valeurs qui peuvent éventuellement être atteintes à partir d'un régresseur considéré à un instant donné, avec les probabilités qu'elles se réalisent effectivement.

5.7 Modèles RBFNs locaux

Pour chacune des N_{out} classes de la carte *OUT*, on modélise les données de la classe par un réseau de type RBFN. L'apprentissage du RBFN est réalisé localement en considérant les $p+q$ composantes (de la variable explicative et de la variable exogène) du régresseur étendu $X'(t)$ comme entrée du réseau RBFN, et la dernière composante $x(t+1)$ de la variable explicative comme sortie du réseau RBFN. Ces modèles RBFNs locaux représentent l'évolution locale de la série à l'intérieur des classes de la carte *OUT*.

5.8 Prédiction

Après toutes ces manipulations, un modèle des données est disponible pour prédire la série.

Ayant à disposition les données jusqu'à l'instant n, on souhaite donc connaître la valeur à l'instant $n+1$. La valeur fournie par le modèle sera une estimation de la valeur réelle, notée $\hat{x}(n+1)$. Cette estimé $\hat{x}(n+1)$ est obtenue comme suit.

On calcule le régresseur $X(n)$. Ce régresseur est présenté à la carte *IN*, et on détermine à quelle classe il appartient. Autrement dit, on recherche quel est le vecteur code de la carte de Kohonen le plus proche. Soit IN_k ce vecteur code, $1 \leq k \leq N_{in}$.

Dans la table des fréquences, on observe la $k^{\text{ième}}$ ligne. Pour cette ligne, on sélectionne les colonnes non nulles. Le régresseur $X(n)$ est alors présenté aux modèles

RBFNs correspondant à ces classes non nulles. Ces colonnes représentent des évolutions possibles de la série quand on se trouve à l'instant *n*.

Chacun des modèles RBFNs donne une sortie correspondant à $X(n)$, sortie notée $\hat{x}_j(n+1)$; cette dernière est une prédiction locale obtenue pour chaque classe *j* de la carte *OUT* dont le coefficient $F(k, j)$ est non nul dans la table, $1 \leq j \leq N_{out}$. L'estimation finale de la valeur suivante est obtenue en pondérant les prédictions locales en fonction des fréquences $F(k, j)$:

$$\hat{x}(n+1) = \sum_{j=1}^{N_{out}} F(k, j) x_j(n+1). \tag{5}$$

6 RÉSULTATS EXPÉRIMENTAUX

6.1 Méthodologie

Afin de tester la méthode de prédiction présentée dans la section précédente, l'ensemble de données est divisé en trois parties : l'ensemble d'apprentissage, l'ensemble de validation, et l'ensemble de test.

Pour deux valeurs N_{in} et N_{out} fixées, l'ensemble d'apprentissage permet de déterminer la position des vecteurs codes à l'intérieur de la distribution des régresseurs $X(t)$ et des régresseurs étendus $X'(t)$, ainsi que les paramètres des différents RBFNs (centres, largeurs, et poids respectifs).

L'ensemble de validation est utilisé pour déterminer les valeurs optimales de N_{in} et N_{out}, de même que le nombre de fonctions Gaussiennes à utiliser dans les réseaux RBFNs. Ces nombres peuvent être optimisés en comparant les performances mesurées par un critère d'erreur de prédiction. Ici, il a été choisi d'utiliser le critère de l'erreur absolue moyenne (Mean Absolute Error) défini par :

$$MAE = \frac{1}{\#V_{set}} \sum_{t=0}^{\#V_{set}-1} \left\| x(t+1) - \hat{x}(t+1) \right\|. \tag{6}$$

Pour pouvoir interpréter plus facilement les résultats obtenus, ce critère a été modifié et normalisé par l'erreur *MAE* qui aurait été obtenue par un modèle de marche aléatoire :

$$NMAE = \frac{\sum_{t=0}^{\#V_{set}-1} \left\| x(t+1) - \hat{x}(t+1) \right\|}{\sum_{t=0}^{\#V_{set}-1} \left\| x(t+1) - x(t) \right\|}. \tag{7}$$

Ce second critère a une interprétation intuitive immédiate : une *NMAE* proche de 1 signifie que le modèle ne modélise pas mieux la série que ne l'aurait fait un modèle de marche aléatoire. Par contre, une valeur proche de 0 indique que le modèle est nettement plus performant en prédiction qu'un modèle de marche aléatoire Gaussienne.

Enfin, lorsque les nombres N_{in} et N_{out} ont été optimisés sur l'ensemble de validation, on évalue les capacités de prédiction de ce modèle optimal sur l'ensemble de test, en mesurant, grâce au *NMAE*, l'écart entre les prédictions et les vraies valeurs pour cet ensemble.

6.2 Les données

La méthode a été appliquée à la prédiction de l'indice allemand DAX30. La période considérée s'étend du 2 janvier 1992 au 8 janvier 2003. La figure 1 représente la série temporelle brute, dans sa partie supérieure, ainsi que les rendements de cette série, dans la partie inférieure.

Outre cette série, la série des volatilités implicites du DAX30, notée par la suite VDAX30, est également présentée, dans la figure 2, partie supérieure, avec en partie inférieure les rendements de cette série VDAX30, pour la même période. Les volatilités implicites constituent une mesure de l'anticipation du comportement des marchés par les opérateurs, à partir de leur intégration de toutes les informations disponibles. Ces volatilités implicites sont les volatilités qui devraient être utilisées dans un modèle Black-Scholes afin d'obtenir le prix des options observé sur le marché. [7].

Dans les régresseurs et les régresseurs étendus qui ont été construits lors de l'application de la méthode à la prédiction du DAX30, les éléments $x(t)$ correspondent à des valeurs passées du DAX, alors que les éléments $u(t)$ correspondent à des valeurs passées du VDAX. Dans un premier temps, la méthode a été appliquée telle quelle. Ensuite, il sera présenté les résultats obtenus en ajoutant de l'information exogène, en utilisant comme variable exogène une prédiction du VDAX30.

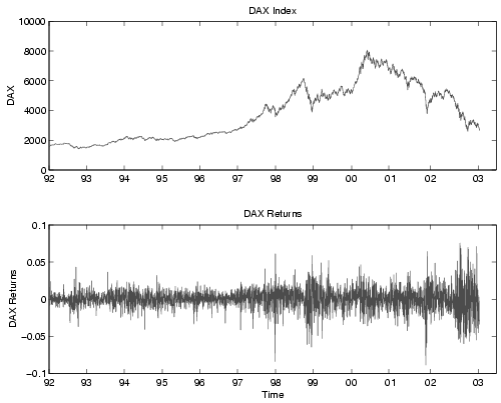

Figure 1 : la série temporelle du DAX30 (au-dessus) et de ses rendements (en dessous), entre le 2 janvier 1992 et le 8 janvier 2003.

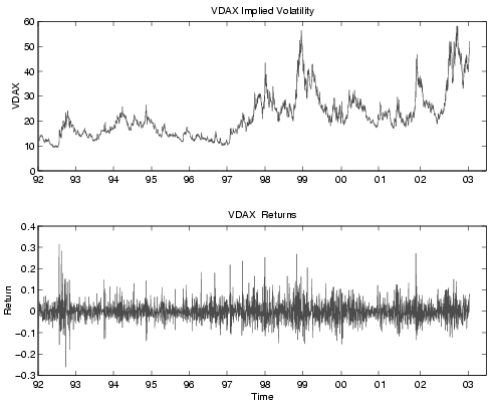

Figure 2 : les séries temporelles du VDAX30 (au-dessus) et des rendements (en dessous), pour la même période.

6.3 Résultats de la prédiction de l'indice DAX30

Dans cette première simulation, les régresseurs construits sur base des relations (1) et (3) contiennent des données DAX30 comme variables explicatives et des données VDAX30 comme variables exogènes. Ces données ont été divisées en ensembles d'apprentissage, de validation et de test, à hauteur de 80%, 10% et 10% respectivement. Plusieurs modèles ont été testés, avec un nombre de vecteurs codes croissant sur chacune des deux cartes *IN* et *OUT*. Le modèle retenu, qui a la plus petite *MAE* sur l'ensemble de validation, utilise deux cartes carrées comportant 4x4 vecteurs codes, aussi bien pour la carte *IN* que pour la carte *OUT*. La figure 3 présente les régresseurs utilisés en apprentissage, regroupés dans les différentes classes de la carte *IN*. La figure 4 représente les vecteurs codes de ces classes. La figure 5 présente les régresseurs étendus utilisés en

apprentissage, eux aussi regroupés dans les différentes classes de la carte *OUT*. La figure 6 représente les vecteurs codes correspondant à ces classes de la carte *OUT*.

Concernant les modèles locaux RBFNs, le nombre de fonctions Gaussiennes utilisées dans chacune des classes de la carte *OUT* a été sélectionné sur base de l'ensemble de validation. Ce nombre est différent pour chaque classe, mais il varie toujours entre 5 et 8 fonctions Gaussiennes.

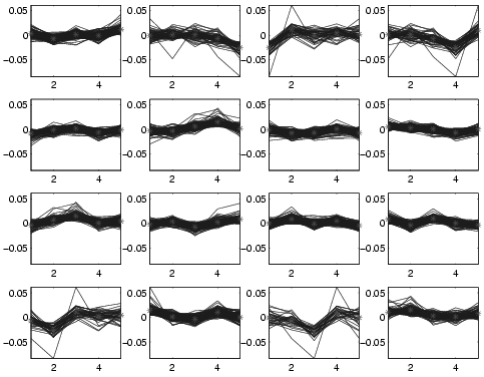

Figure 3 : Les régresseurs $X(t)$ utilisés pour l'apprentissage, répartis dans les classes de la carte *IN*.

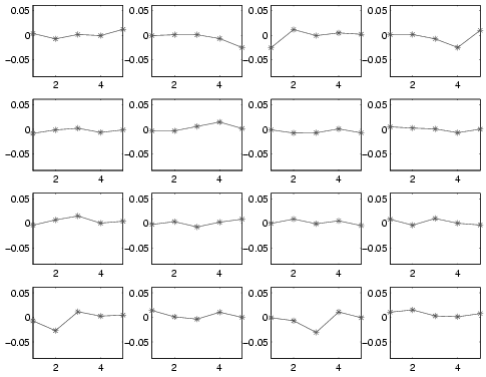

Figure 4 : Les vecteurs codes correspondants aux classes de la carte *IN*.

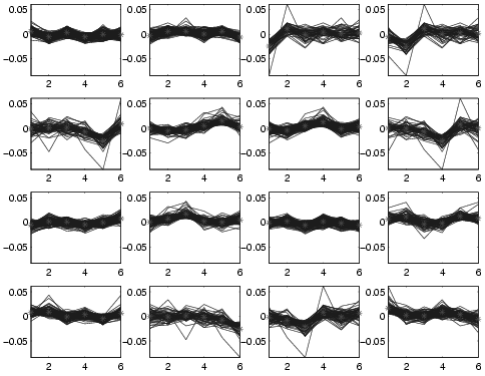

Figure 5 : Les régresseurs étendus $X'(t)$ utilisés pour l'apprentissage de la carte *OUT*, répartis dans les différentes classes de la carte.

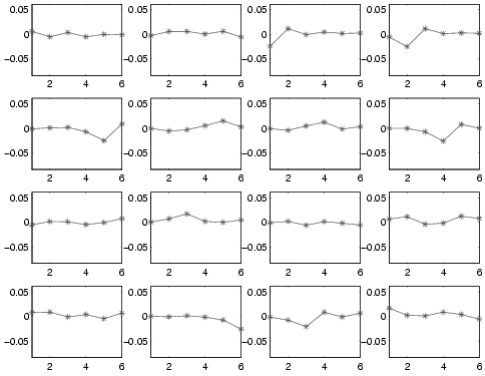

Figure 6 : Les vecteurs codes correspondants aux classes de la carte *OUT*.

Le tableau 1 présente un exemple de table de fréquence. Ce tableau reprend en fait une partie de la table 16x16 obtenue sur les données du DAX30. Si on regarde maintenant la première ligne de la table par exemple, on peut y lire que, si un régresseur appartient à la première classe de la carte *IN*, alors le régresseur étendu lui correspondant se retrouve dans 95% des cas dans la classe 11 de la carte *OUT*, et dans 5% des cas dans la classe 15. Si un régresseur appartient à la deuxième classe de la carte *IN*, alors le régresseur étendu lui correspondant se situe dans 24% des cas dans la classe 2 de la carte *OUT*, dans 13% dans la classe 4, etc.

Afin de pouvoir évaluer les performances de la méthode, l'erreur NMAE obtenue est comparée à deux modèles de références. Le premier modèle de référence est le modèle de marche aléatoire dont l'erreur *NMAE* est, bien entendu, de 1. Le second modèle de

référence est un modèle RBF global, dont les paramètres ont été fixés par apprentissage sur le même ensemble d'apprentissage que pour la méthode SOMS et RBFNs locaux, les régresseurs étant identiquement construits. Le choix d'un modèle RBFN est motivé par le fait que la littérature indique qu'un modèle RBFN semble être équivalent aux meilleurs modèles économétriques [8, 9, 10]. Pour le modèle RBFN global, plusieurs modèles avec un nombre croissant de fonctions Gaussiennes ont également été testés. Celui retenu pour la comparaison est celui dont l'erreur MAE sur l'ensemble de validation est la plus petite. Le tableau 2 donne les valeurs de l'erreur NMAE obtenues sur l'ensemble de test pour ces deux modèles de référence et le modèle SOMS et RBFNs locaux.

On peut observer que le modèle SOMS et RBFNs locaux donne une erreur NMAE inférieure à celle obtenue par les deux modèles de référence. Ce constat peut s'expliquer par la particularité de la méthode qui permet de séparer les différentes dynamiques sous-jacentes de la série (par les SOMs) et de les modéliser séparément (par les RBFNs locaux).

0	0	0	0	0	0	0	0	0	0	.95	0	0	0	.05	0
0	.24	0	.13	.03	.13	0	.01	0	.39	0	.05	.01	.02	0	0
0	0	.95	0	0	.03	0	0	0	0	0	.03	0	0	0	0
.82	0	0	0	0	0	0	0	.14	0	0	0	.04	0	0	0
0	0	0	.09	0	0	0	.91	0	0	0	0	0	0	0	0
0	.03	0	0	.79	0	.06	0	0	0	.01	.03	0	.09	0	0
0	.01	0	.6	.01	0	.09	0	.01	.02	0	0	.01	.24	.01	0
.15	0	.03	.06	.01	0	.21	0	.36	0	0	.07	.06	.06	0	0
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

Tableau 1 : Extrait de la table des fréquences : moitié supérieure de la table 16x16 du meilleur modèle obtenu sur l'ensemble de validation.

	Marche aléatoire	RBFN global	SOMs et RBFNs
NMAE	1	0.674	0.454

Tableau 2 : Comparaison des résultats de l'erreur *NMAE* obtenus en prédiction avec différents modèles sur l'ensemble de test (10% des données, soit 288 données).

6.4 Utilisation de la prédiction du VDAX30 pour prédire le DAX30

Pour cette deuxième simulation, les régresseurs ont été construits en y ajoutant une composante de la variable exogène VDAX30, à savoir la valeur suivante en t+1, ou plutôt la prédiction de celle-ci (bien entendu : en t, la vraie valeur en t+1 est inconnue). En effet, la volatilité implicite constitue une mesure de l'anticipation du comportement du marché. Connaître la volatilité implicite à l'instant t+1 permet donc de mieux prédire la valeur de la série en t+1, puisqu'on inclut déjà le comportement futur du marché. En suivant cette intuition, un modèle a été développé, pour permettre de prédire le VDAX à l'instant t+1. Cette prédiction a été intégrée dans les régresseurs, et la méthode SOMS et RBFNs locaux a ensuite été appliquée à ces nouveaux régresseurs. Comme pour la simulation précédente, les données ont été divisées en apprentissage, validation et test.

Le tableau 3 présente une comparaison des résultats obtenus avec le modèle SOMS et RBFNs locaux, en utilisant et sans utiliser la prédiction du VDAX30. Deux modèles de prédiction du VDAX30 ont été développés. Le premier est un RBFN global ; le second est la méthode proposée dans cet article. Dans ce deuxième cas, mentionnons que les

cartes de Kohonen sont cette fois des cartes carrées 3x3, et quel les modèles RBFNs comptaient chacun 4 fonctions Gaussienne.

	SOMs et RBFNs sans prédiction VDAX30	SOMs et RBFNs avec prédiction VDAX30 par RBFN global	SOMs et RBFNs avec prédiction VDAX30 par SOMs et RBFNs
NMAE	0.454	0.308	0.293

Tableau 3 : Impact de l'utilisation de la prédiction de la valeur suivante du VDAX30 comme variable exogène pour prédire le DAX30 : comparaison des résultats de l'erreur *NMAE* obtenue en prédiction avec différents modèles sur l'ensemble de test.

7 CONCLUSION

Dans cet article, une méthode générale de prédiction de séries temporelles a été présentée. Cette méthode a été développée spécifiquement pour le cas des séries comportant plusieurs dynamiques sous-jacentes, et semblent donc particulièrement bien adaptée au problème de la prédiction de séries financières.

Deux cartes de Kohonen sont utilisées pour partitionner respectivement le passé de la série et son évolution. Chacune des zones définies par la seconde carte de Kohonen est modélisée par un réseau RBFN local. Ces réseaux locaux permettent chacun d'obtenir une prédiction locale ; celles-ci sont fusionnées en une seule prédiction globale au moyen d'un modèle stochastique issu d'une modélisation de la distribution empirique des données à l'intérieur des zones définies par les cartes de Kohonen.

Cette méthode a été appliquée sur la série du DAX30. Il a par ailleurs été montré dans quelle mesure l'extension du modèle pour prendre en compte des variables exogènes supplémentaires permet d'améliorer les résultats jusqu'alors obtenus. Ces résultats sont prometteurs, et surpassent assez clairement ceux obtenus avec un modèle de marche aléatoire appliqué à cette même série.

8 REMERCIEMENTS

Le travail de G. Simon est soutenu par une bourse du F.R.I.A. M. Verleysen est Maître de Recherche au F.N.R.S. Le travail d'A. Lendasse est soutenu par le Pôle d'Attraction Interuniversités (IAP), créé sur l'initiative de l'Etat Fédéral Belge, Ministère des Sciences, des Technologies et de la Culture. La responsabilité scientifique n'engage que les auteurs.

9 BIBLIOGRAPHIE

[1] F. Diebold, J.-H. Lee, G. Weinbach (1994), Regime switching with time-varying transition probabilities, in non stationary time series analysis and cointegration, C. Hargreaves editors, Oxford University Press, pp 283-302,.
[2] M. Verleysen, E. de Bodt, A. Lendasse (1999), Forecasting financial time series through intrinsic dimension estimation and non-linear data projection, in Proc. of International Work-conference on Artificial and Natural Neural networks (IWANN'99),

Springer-Verlag Lecture Notes in Computer Science, n° 1607, pp. II596-II605, June 1999.
[3] T. Kohonen (1997), Self-Organizing Maps, Springer Series in Information Sciences, Vol. 30, 2nd edition.
[4] S. Haykin (1999), *Neural Networks - A comprehensive foundation*, 2nd edition, Prentice Hall.
[5] M. J. Orr (1998), Optimising the Widths of Radial Basis Functions, in Proc. of Vth Brazilian Symposium on Neural Networks, Belo Horizonte, Brazil, December 1998.
[6] N. Benoudjit, C. Archambeau, A. Lendasse, J. Lee, M. Verleysen (2002), Width optimization of the Gaussian kernels in Radial Basis Function Networks, in Proc. of 10th European Symposium on Artificial Neural Networks (ESANN'2002), d-side, Brussels, Belgium, April 2002.
[7] J.C. Hull (2000), Options, futures, and Other Derivatives, Prentice Hall, 5th edition.
[8] A.P. Blake, G. Kapetanios (2000), A radial basis function artificial neural network test for ARCH, Economics Letters, n° 69, pp 15-23.
[9] P.H. Franses, D. van Dijk (2000), Nonlinear Time series models in Empirical Finance, Cambridge University Press.
[10] A.P. Refenes, A.N. Burgess, Y. Bentz (1997), Neural Networks in Financial Engineering: A Study in Methodology, IEEE transactions on Neural networks, Vol. 8, n°. 6, pp.1222-1267, November (1997).