Système de trading automatique et prédiction de cours à l'aide de réseaux de neurones

Odyssée Tremoulis

CONSERVATOIRE NATIONAL DES ARTS ET METIERS PARIS

Mémoire présenté en vue d'obtenir le

Diplôme d'ingénieur en informatique

option Architecture et Ingénierie des Systèmes et des Logiciels

SYSTÈME DE TRADING AUTOMATIQUE ET PRÉDICTION DE COURS À L'AIDE DE RÉSEAUX DE NEURONES

Par

TREMOULIS, Odyssée

Soutenu le 15 Mai 2012

JURY : François-Yves VILLEMIN, Patrice LIGNELET

PRESIDENT : Yann POLLET

MEMBRES : N. WAUTERS (Thales), A. DE CHAMBOURCY (Redsen)

RÉSUMÉ

Avec l'avènement du trading algorithmique sur les places financières mondiales, de nouvelles technologies voient le jour dans les institutions financières. Ce mémoire propose une étude sur une des catégories de ce type de trading : le trading haute fréquence.

Après y avoir détaillé les fonctions et les stratégies utilisées par cette technologie, nous mettons au point le prototype d'un tel type de système permettant d'investir de manière automatique sur les marchés. Nous y fournissons les spécifications fonctionnelles et techniques en mettant l'accent sur une particularité du système : les réseaux de neurones. Enfin, nous y présentons des premiers résultats d'investissement fictif.

MOT CLÉS

Bourse, courtier, marchés financiers, fonds d'investissements, analyse quantitative, analyse technique, analyse fondamentale, trading électronique, trading algorithmique, algotrading, trading haute fréquence, protocole Fix, ratio de perte maximale, stratégies d'investissement, réseau de neurones, perceptron, perceptron multicouches, feedforward, retro propagation, reconnaissance de modèles, prédiction, sigmoïde, moyenne absolue de l'erreur, symétrie directionnelle, gestion du risque, value at risk.

ABSTRACT

With the advent of algorithmic trading on world financial markets, new technologies are emerging in financial institutions. This thesis describes a study of such trading fields: the high frequency trading.
After detailing the functions and strategies used by this technology, we develop a prototype of this kind of system to automatically invest in the markets. We provide the functional and technical specifications focusing on a particular system: neural networks. Finally, we present the first results of fictitious automated investments.

KEY WORDS

Exchange, broker, financial markets, investment funds, quantitative analysis, technical analysis, fundamental analysis, electronic trading, algorithmic trading, algotrading, high frequency trading, fix protocol, ratio of maximum loss, investment strategies, neural networks, perceptron, multilayer perceptron, feedforward, backpropagation, pattern recognition, prediction, sigmoid, mean absolute error, directional symmetry, risk management, value at risk.

SOMMAIRE

TABLE DES FIGURES

ABRÉVIATIONS

Abréviations	Signification
THF	Trading Haute Fréquence
MLP	Multi Layer Perceptron (Perceptron multi couches)
HONN	High Order Neural Network (Réseau de neurones à haut ordre)
NN - RN	Neural Network - Réseau de neurones

GLOSSAIRE DES TERMES TECHNIQUES

Termes	Définition
Broker - Courtier	Intermédiaire entre une place boursière et un client

1. INTRODUCTION

« En 2011, 40% des ordres donnés sur le CAC40 sont totalement automatisés à l'aide d'algorithmes informatiques. »

Depuis la création de la finance de marché, l'utilisation des moyens humains pour investir sur les marchés diminue d'année en année pour être remplacée par des systèmes informatiques plus fiables que l'homme, appelé « *Algotrading* » (contraction de « trading algorithmique »). Ceci est dû au fait que les traders peuvent prendre plusieurs jours (voire semaines) avant de réaliser des bénéfices et cela implique de forts risques comme le cas Kerviel à la Société Générale. Le trading algorithmique mise sur le passage d'une grande quantité d'ordres de bourse à une haute fréquence réalisant une multitude de petits gains. D'autre part, les institutions financières voient, dans la course au profit, que le trader (être humain) est particulièrement limité pour investir sur les marchés : il ne peut assimiler qu'une quantité d'informations limitée, il manque de réactivité et n'assure pas des ordres 24/24h sur différentes places boursières. Elles n'hésitent donc plus à investir dans d'énormes plateformes informatiques pour remplacer ces traders. Elles ne cherchent pas à reproduire l'intelligence humaine mais à tirer parti des faiblesses des marchés boursiers pour en produire des bénéfices.

Ce mémoire présente et étudie un type de plateforme de trading automatique. Il se découpe en deux parties : La première fait un état des lieux des systèmes de trading actuels. La deuxième est dédiée à la spécification, au développement et à l'optimisation d'un système de trading automatique. Nous fournissons en plus une étude sur l'intelligence artificielle et la façon dont nous l'utiliserons dans notre système.

1.1. OBJECTIF

L'objectif de ce mémoire est d'étudier, de développer et d'exploiter un système informatique permettant d'investir sur les marchés financiers de façon automatique (sans intervention humaine).

Ce système analysera les cours de bourse des marchés financiers en temps réel à l'aide de stratégies (analyse technique, quantitative, …) que nous aurons développé au préalable. Puis il évaluera le risque futur grâce à une intelligence artificielle capable de

faire une prédiction de l'avenir des cours boursiers. Ainsi le système sera en mesure de prendre la décision adéquate à un instant donné, d'investir ou au contraire, de se retirer du marché.

Nous développerons un prototype nous permettant de valider notre conception du système. Ce prototype contiendra le minimum de fonctionnalités comme un éventail de stratégies et une intelligence artificielle. Nous pourrons évaluer son fonctionnement avec l'historique des cours de bourse.

La validation de ce prototype se fera par sa cohérence technique : le système est-il capable de fonctionner dans un environnement temps réel et est-il facilement évolutif ? Puis par son niveau de profitabilité : permet-il de réaliser des bénéfices sur l'historique des cours de bourse ?

2. LE TRADING AUTOMATIQUE

Dans ce chapitre, nous détaillons une des méthodes les plus courantes de trading algorithmique puis nous y décrirons les stratégies d'investissement les plus repandues dans le domaine du trading.

2.1. ÉTAT ACTUEL

Selon Irene Aldrige (2010) nous pouvons distinguer 3 familles de trading automatique qui chacune se succèdent au fil de l'évolution des moyens des entreprises:

- Le trading électronique. Cette famille représente les passages d'ordre par téléphone comme le faisaient les anciens traders et les nouvelles méthodes d'exécution d'ordre par ordinateur.
- Le trading algorithmique qui consiste en l'optimisation d'ordres exécutés par les traders. En réalité, le trader reste maître de la décision d'investir ou pas, mais c'est la technique d'investissement en aval qui est automatisée (comme le passage d'un ordre à 10 millions d'euros qu'il faut découper en petits lots afin de ne pas impacter le prix d'achat/vente)
- Le trading haute fréquence (THF) qui consiste en la décision et l'exécution totalement automatique de passage d'ordre sur le marché à très haute fréquence (achat-vente en quelques secondes)

La première famille a été une évolution générale des systèmes de trading. L'avènement des réseaux et des ordinateurs fait qu'elle est aujourd'hui indispensable à toute communication inter établissements. Puis le trading algorithmique s'est rapidement généralisé au fur et à mesure que les investissements devenaient de plus en plus importants. Il a toutefois le désavantage de faire appel à des *hommes* et nécessite donc des moyens financiers importants pour gérer ces ressources humaines. C'est pourquoi notre intérêt portera sur la dernière famille. Cette dernière famille est particulièrement intéressante car d'une part, les techniques actuelles sont encore jeunes (elle est utilisée à New York que depuis 10 ans), et d'autre part, elle nécessite des moyens techniques abordables si l'on souhaite en avoir une utilisation à petite échelle. C'est pourquoi nous porterons notre étude sur cette famille-là. Du fait qu'elle représente le nouveau domaine

de compétition des institutions financières, l'information disponible sur les techniques de trading haute fréquence est peu divulguée.

Voici un graphe représentant les raisons principales pour lesquelles les institutions financières ont recours au THF :

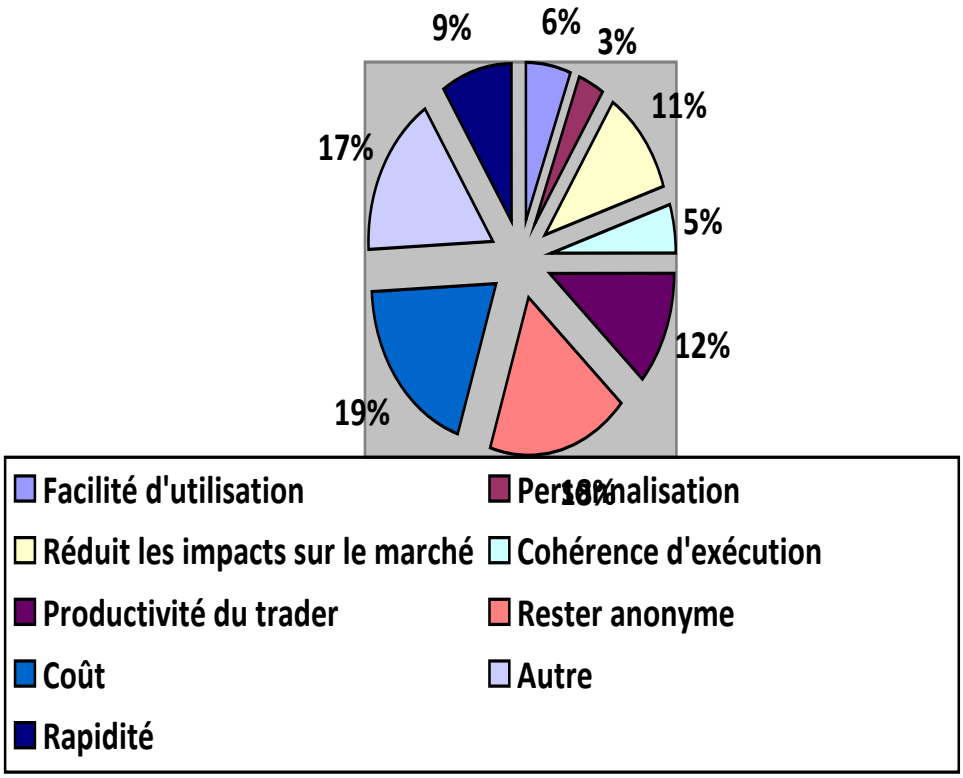


Figure 1 - Raisons d'utilisation de l'algotrading d'après "The TRADE Annual Algorithmic Survey"

Ce graphe nous montre que les principales raisons d'utilisation de l'algotrade sont par ordre d'intérêt:

- La facilité d'utilisation.
- Le coût. Comme nous le disions, aucune ressource humaine n'est à gérer, donc une importante réduction des coûts.

- Rester anonyme. Cela donne la possibilité aux gros investisseurs de cacher leurs intentions d'investissement en réalisant de multiples exécutions de petites tailles au cours d'une période.

2.2. LE TRADING HAUTE FRÉQUENCE

Les établissements financiers sont à la recherche de moyens toujours plus complexes pour atteindre leurs objectifs. Un de ces moyens est le Trading Haute Fréquence « THF » (de l'anglais « High Frequency Trading »).

Le THF est un ensemble de super-ordinateurs et d'algorithmes informatiques capable de passer des ordres sur les marchés en quelques microsecondes. Ainsi, cela permet à ces super-ordinateurs d'avoir une longueur d'avance sur les traders et par conséquent de réaliser des bénéfices en réduisant au maximum le risque.

LE THF DANS LE MONDE

Le THF nécessite la mise en place d'importants moyens financiers. De ce fait il n'est accessible qu'aux grosses institutions financières.

Représenté sur chaque grande place boursière, le THF est source de:

- 73% des ordres sur les marchés des États-Unis en 2009 d'après Rob lati (2009)
- 40% des ordres sur les marchés français en 2011.

Dorénavant, de plus en plus de courtiers pour particuliers, mettent à disposition leur technologie pour permettre à leurs clients d'élaborer eux-mêmes leur stratégie.

STRATÉGIES

Une stratégie est un ensemble de modèles mathématiques utilisé pour détecter un comportement dans un cours de bourse qui indiquera la tendance et donc les positions à prendre pour profiter de manière la plus optimale de la situation.

Deux types de stratégies se sont distingués au fil des années :

L'analyse technique utilisée pour détecter les comportements dans les cours de bourse, ou pour détecter l'état *psychologique* du marché, a prospéré au début du 20éme siècle. Utilisée encore fréquemment aujourd'hui, elle sert à passer des ordres à la journée pour une durée minimum de quelques jours.

L'analyse fondamentale est une analyse basée sur l'information fondamentale d'une entreprise c'est-à-dire la gestion et les avoirs de celle-ci. Elle prend en compte toutes les données comptables de l'entreprise, ses brevets (…) et fait une estimation de sa valeur. Elle part du principe que si le prix d'une action diverge du prix établi à l'aide de ces informations, il ne pourra qu'y revenir.

Certaines parties de l'analyse fondamentale sont utilisées aujourd'hui dans le trading haute fréquence. Par exemple, l'arbitrage consiste à profiter de la différence entre une même action cotée sur deux places boursières différentes. Si l'action l'Oréal monte en France, l'écart avec l'action cotée à New-York ne mettra que quelques secondes à monter. Ainsi l'action tend toujours vers sa valeur de référence.

Aujourd'hui, ces deux types de stratégie ne suffisent plus.

Les physiciens et mathématiciens recrutés à la fin du siècle dernier pour établir les modèles mathématiques des stratégies vues précédemment, ont laissé place au « Quant trader ». Ces nouveaux analystes quantitatifs que les fonds d'investissement s'arrachent, sont les nouveaux créateurs des stratégies d'aujourd'hui, basées sur des modèles statistiques et des principes scientifiques.

Là où l'analyse technique démontre, quand un cours est trop haut ou trop bas, qu'il va y avoir un retournement de tendance, et là où l'analyse fondamentale démontre qu'un cours doit se rapprocher de sa valeur fondamentale, l'analyse quantitative a l'avantage de ne pas prendre de décisions discriminatoires, c'est-à-dire qu'elle se base uniquement sur les effets du marché sans prendre (ou très peu) en compte les évènements extérieurs ou passés.

Donc, dans ce type de système tout repose sur le placement du signal (achat/vente). Le signal donnera le top départ pour un achat ou une vente. Un mauvais signal peut rapidement tourner un investissement en une grosse perte. Il sera évident que les calculs et les ordres devront être réalisés en une fraction de seconde afin que le marché ne change pas d'état entre le moment de prise de décision et le moment d'exécution.

Selon I. Aldrige (2010) les stratégies peuvent être classées en 4 grandes catégories :

Stratégie	Description	Temps de garde typique
Fourniture de liquidité automatisée	Algorithmes quantitatifs pour un prix optimal et l'exécution des positions sur le marché	< 1 minute
Microstructure des marchés boursiers	Identifier les flux d'ordre en faisant de l'ingénierie inversée sur les prix constatés	< 10 minutes
Trade événementiel	Trade à court terme sur des macros évènements	< 1 heure
Déviation d'arbitrage	Arbitrage statistique des déviations de l'équilibre : trade en triangle, trade basique et les trades similaires…	< 1 jour

Deux techniques classiques du trading à haute fréquence

Une des premières techniques des traders est d'évaluer le prix de l'action c'est-à-dire réussir à estimer à combien les investisseurs sont-ils prêts à payer pour cette action.

Lorsque le trader se retrouve avec quelques dizaines de milliers d'actions à vendre, il ne peut pas tout mettre en vente d'un seul coup. La quantité étant trop importante, il y aurait un impact sur le marché, donc une modification du marché. Cela a deux inconvénients : il peut s'agir de manipulation de marché et dans ce cas, l'autorité des marchés boursiers (AMF) peut sanctionner. Le deuxième inconvénient est qu'au fur et à mesure de l'achat d'action, cela crée automatiquement de la demande, donc les prix montent. Entre le prix de la première action achetée et le prix de la dernière, il peut y avoir un écart significatif. Donc il va découper cette vente en petits lots et estimer à combien se vend le premier lot. Puis il adaptera les lots d'actions suivants en fonction de la demande.

Le THF va user de sa rapidité. Il va réaliser la même opération que le trader : une découpe du lot d'actions en plus petits lots. Il va demander au système boursier de

placer son ordre un peu en dessous du prix actuel. Avant que ce premier lot d'action soit vendu, il le retirera du marché. Cette technique va lui permettre d'évaluer le prix de l'action par rapport au comportement du marché. Ainsi, il aura réussi à évaluer le prix réel de l'action sans vendre. À partir de ce moment, le système va pouvoir prendre position en connaissant le prix futur de l'action. Cette technique exploite la lenteur des autres acteurs du marché.

Une deuxième technique consiste à réaliser des « ordres flashs ». Lors du passage d'un ordre, des informations concernant l'ordre sont transmises du système boursier au fonds d'investissement une fraction de seconde avant la validation de cet ordre. Ainsi le système de THF passe un ordre non significatif afin de connaître les informations sur le cours et ainsi passer un second ordre avec un prix de l'action mieux adapté à l'offre et la demande.

Cette technique exploite aussi la lenteur des autres acteurs du marché en ayant accès à des informations avant eux.

2.3. LES STRATÉGIES DE TRADING ALGORITHMIQUE

Nous présentons ici les stratégies de trading les plus utilisées par les traders et les systèmes d'algotrading. Ces stratégies vont nous guider pour l'élaboration de notre système.

SUIVRE LA TENDANCE
Cette stratégie profite d'une tendance du marché pour prendre position et profiter d'un mouvement régulier à la hausse ou à la baisse d'un cours.

ÉCHANGE DE PAIRES
Ici, le système va profiter d'un écart entre deux actions qui sont potentiellement corrélées.

Ex : Une chute du cours de Coca-Cola va avoir une incidence sur le cours de Pepsi. Ainsi, lors de la chute du cours de Coca-Cola, nous pouvons prévoir une chute moins importante sur le cours de Pepsi car il existe une corrélation de l'activité entre ces deux entreprises.

ARBITRAGE

Autre cas : Profiter du temps de latence entre deux places boursières qui cotent une même action. Un ajustement du prix de l'action est automatiquement réalisé entre ces deux places boursières.

Ex : L'action BNP Paribas chute sur la place boursière de Paris mais n'a pas encore été impactée à la bourse de Tokyo. Il est donc possible de profiter de ce temps de latence pour acheter à la baisse l'action BNP Paribas à Tokyo.

RÉVISION SIGNIFICATIVE

Cette stratégie part du principe qu'une action a un prix moyen. Quand le prix de l'action est bien supérieur à ce prix moyen suite à une annonce forte sur le marché par exemple, alors il y a surachat ou survente. Cette stratégie est très proche de l'analyse technique et fondamentale.

Ex : Sur les 14 derniers jours, la moyenne du cours de l'action Carrefour est de 20€. Si aujourd'hui elle est à 21€ on peut conclure qu'il y a survente et que le prix va se corriger dans les jours qui viennent.

SCALPING

Cette méthode d'achat vente est élaborée pour faire des profits sur de petites différences. Pour un cours donné, on achète un peu en dessous du cours. Si une contrepartie est prête à vendre à ce prix-là, on revend tout de suite et l'on peut gagner ainsi la différence entre le prix d'achat et le prix de vente. Cela nécessite de réaliser la transaction en moins d'une minute.

Nous avons démontré dans ce chapitre les moyens mis en place par les institutions financières pour améliorer les techniques de trading et détecter automatiquement les opportunités qu'offrent les marchés.

Dans le chapitre suivant, nous allons mettre en place notre propre système de trading. Nous décrirons les différents modules composants ce système et nous le mettrons à l'épreuve sur l'historique des données du marché.

3. SYSTÈME DE TRADING AUTOMATIQUE

Dans ce chapitre nous décrirons le fonctionnement du logiciel d'un point de vue informatique en décrivant son architecture fonctionnelle et ses processus. Puis nous détaillerons la partie technique et présenterons les résultats des premiers tests. Nous appuierons sur la partie du réseau de neurones.

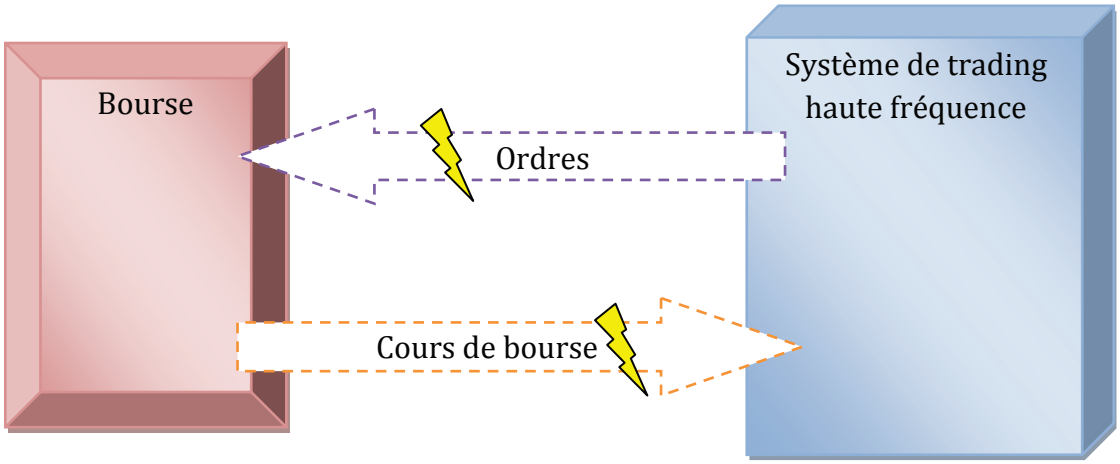


Figure 2 - Schéma global des échanges avec la bourse

La figure 2 présente le schéma des flux globaux entre un système de trading et une place boursière. Nous avons d'un côté la bourse qui envoie les cours à intervalles réguliers au système de trading à haute fréquence puis de l'autre, notre système qui envoie des ordres en fonction de ses stratégies.

Selon I. Aldrige (2010) un système de trading automatique doit assurer les fonctions suivantes :

- *Génération* d'un signal. Elle prend en entrée les cours de bourse, les analyse, génère un signal d'achat/vente et enregistre les pertes et les profits.

- *Aide à l'analyse*. Elle aide à la construction de nouveaux modèles techniques de trading. Cette fonction est souvent assurée par un système extérieur tel que les logiciels MatLab et R.

- *Récupération et analyse de l'information* (rumeurs, nouvelles annonces…). Thomson/Reuters fournit un service d'informations en temps réel dans un format compréhensible par un ordinateur.

- *Optimisation d'exécution*. Elle permet de placer de gros ordres discrètement afin de ne pas révéler les stratégies aux autres investisseurs et d'optimiser les placements à un meilleur prix.

- *Gestion du risque*. Elle permet d'évaluer l'exposition actuelle du marché.

- *Monitoring*. Il sert à l'administration du système, à recevoir des alertes.

- *Recherche.* Elle permet de faire de nouveaux tests, intégrer des informations de sociétés extérieures…

Le schéma suivant présente le système d'un point de vue des processus :

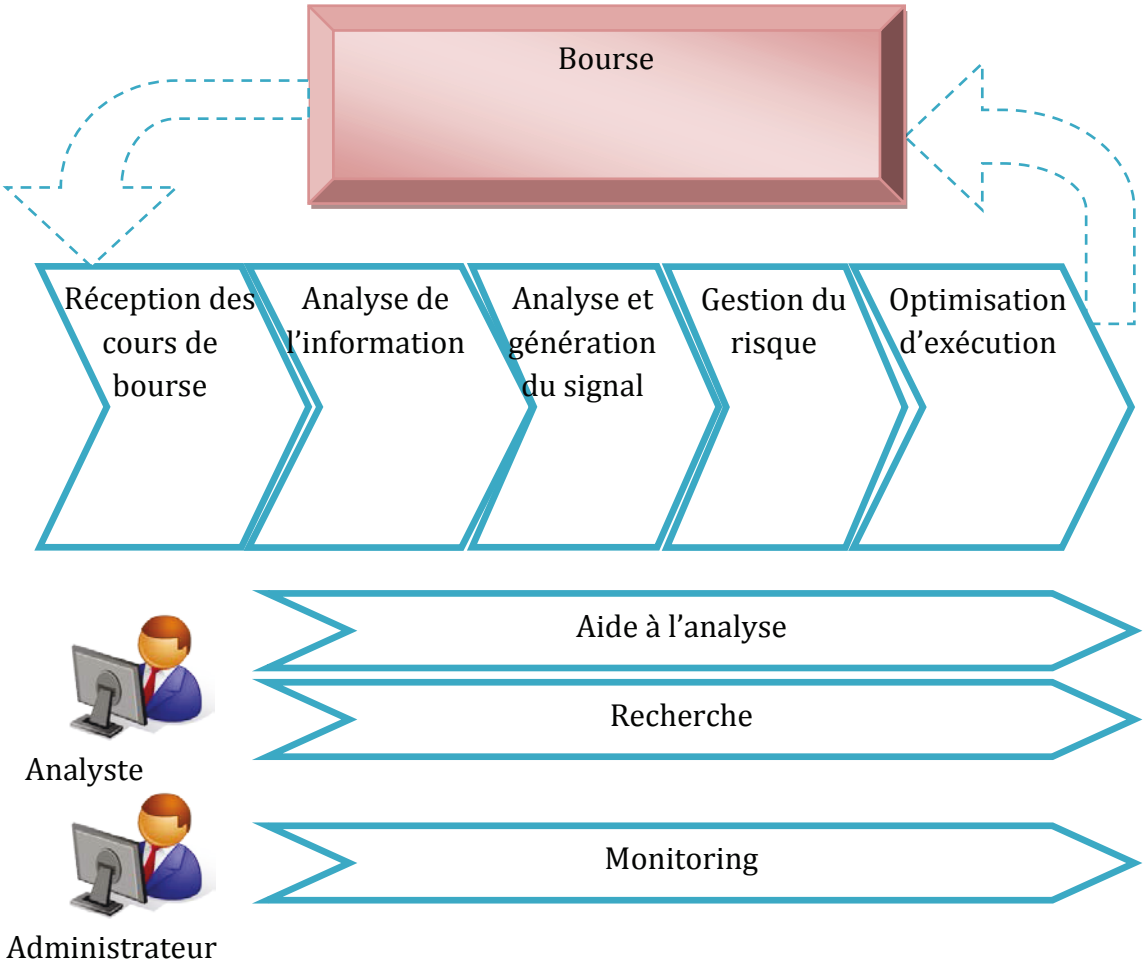


Figure 3 - Diagramme des processus

- Un broker (Institution financière, courtier, place boursière…) qui fournit les cours de bourses et exécute les ordres de bourse.
- Des analystes en charge d'établir de nouvelles stratégies et de faire de la recherche.
- L'administrateur qui a une fonction purement technique en s'occupant de vérifier que le système est opérationnel.

Le flux principal commence par la réception des cours de bourse en temps réel (environ un millier de messages à la seconde) et de l'information boursière (info, news, rumeurs…). Ensuite le système analyse cette information boursière, en déduit les événements possibles qui surviendront à terme (fusion, augmentation de capital, faillite…).

Le troisième processus va se servir des résultats de l'analyse informationnelle pour générer son signal. Il va en plus exécuter toutes les stratégies qui vont lui fournir chacune un signal. Il prendra la décision du signal final en agrégeant tous ces signaux.

Le processus de gestion du risque va prendre en compte le signal du processus précédent et va évaluer le risque si l'on devait le suivre. Ainsi, suivant le seuil de risque défini, le système suivra ou non le signal.

Le dernier processus de cette chaîne permet d'optimiser l'exécution de l'ordre de bourse si cet ordre s'avère trop important à passer. Ex : 10 millions d'euros à découper en lot de mille euros.

Enfin les processus exécutés en parallèle sont la recherche et l'analyse des stratégies gérées par les analystes, et le processus de monitoring géré par l'administrateur.

Dans le sous chapitre suivant, nous présenterons ces processus en détaillant leurs fonctions.

3.1. SPÉCIFICATIONS FONCTIONNELLES

Dans cette partie, nous commençons à élaborer notre système de trading. Nous allons d'abord décrire les fonctions qui le composeront à partir des processus décrits précédemment.

Voici un schéma qui illustre l'organisation des fonctions qui composeront notre système :

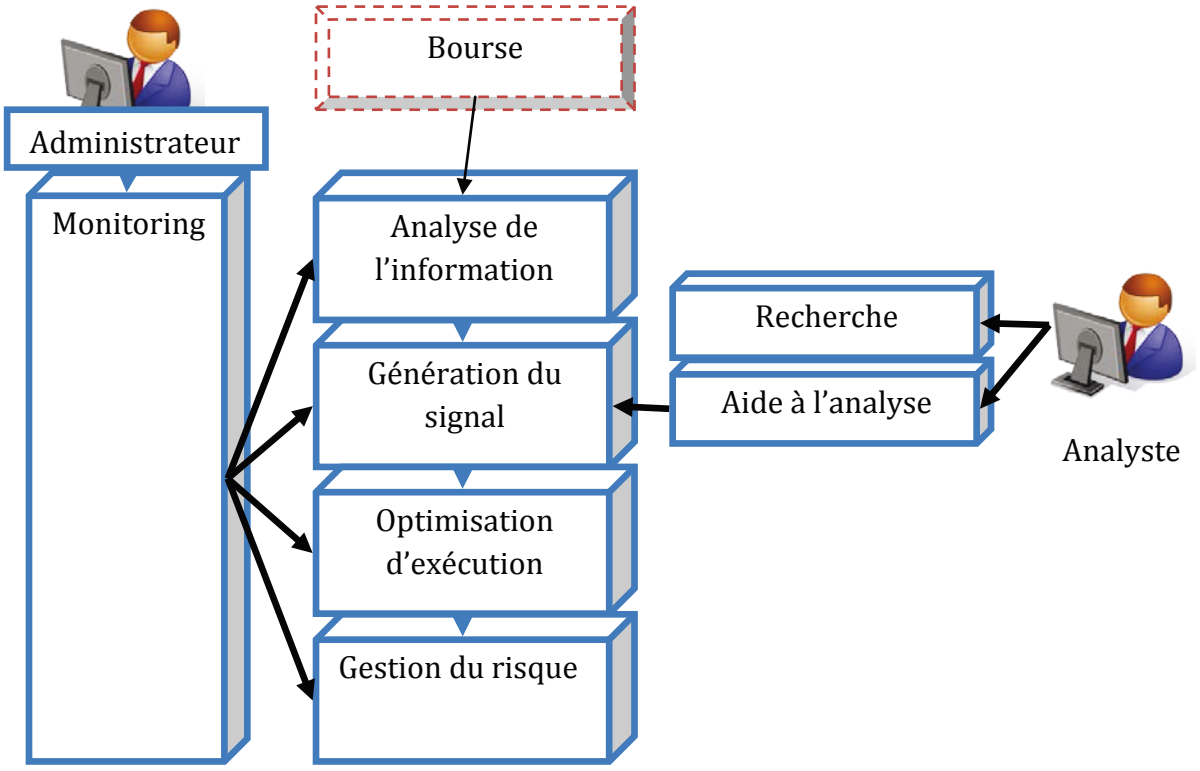


Figure 4 - Vue globale d'un système de trading

Nous y retrouvons nos 7 processus et nos 3 acteurs. Nous ajouterons à ces 7 fonctions deux autres fonctions :

- une *fonction d'interface* qui sera en charge de gérer la communication avec la bourse. Cette fonction comporte, comme nous le verrons plus loin, de multiples sous fonctions. De ce fait, nous ne l'intégrerons pas à une autre fonction.
- une *fonction de réseau de neurones* qui se veut être une évolution plus récente des systèmes de trading actuel.

La fonction d'analyse de l'information ne sera pas traitée dans ce mémoire car les moyens exigés (ex : abonnement à des flux d'informations chez des entreprises spécialisés comme Reuters, Thomson) ne sont pas à notre portée. Elle est généralement accompagnée d'un système de fouille de données.

Également la fonction d'optimisation d'exécution qui, comme nous l'avons vu, permet le découpage de gros investissements en plusieurs lots, ne sera pas traitée car nous ne sommes pas en moyen de fournir d'importants investissements.

Nous allons maintenant détailler chacune de ces fonctions.

3.1.1. *FONCTION DE L'INTERFACE*

L'interface est la première fonction de notre système. Elle sera en charge de :

- *Gérer l'arrivée des cours de bourse*, ce qui inclut de :
    - Demander les cours de bourse à un courtier
    - Gérer la réception de ces cours
    - Transmettre les cours aux autres fonctions

Voici un schéma des flux de gestion des cours avec ces sous fonctions :

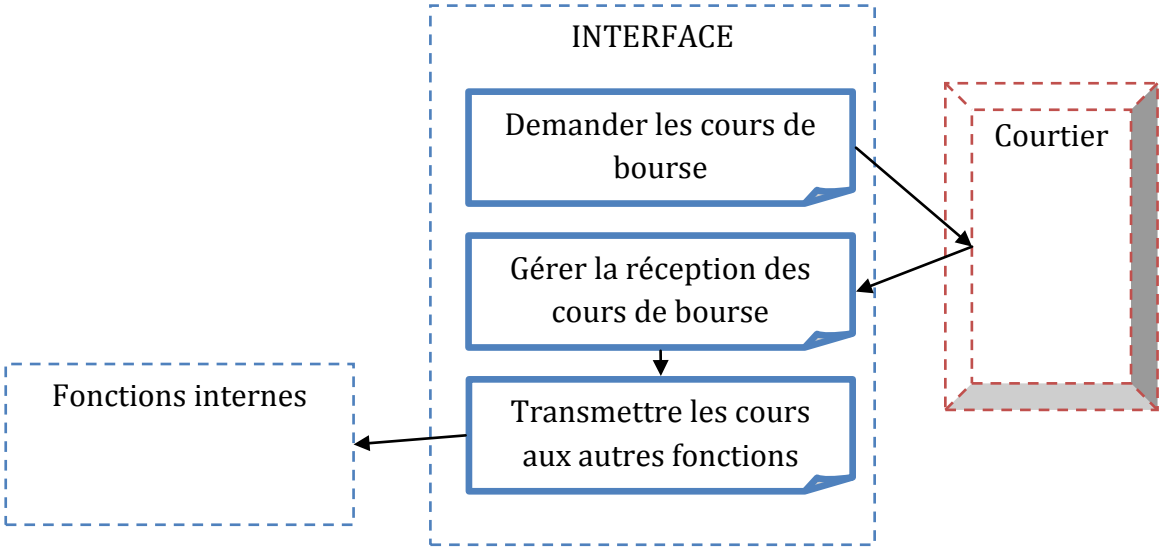


Figure 5 - Diagramme de flux de gestion des cours

- *Gérer la transmission d'ordres*, ce qui inclut de :
    - Gérer la réception des ordres par les fonctions internes.
    - Transmettre les ordres aux courtiers.

Voici le schéma de flux de la sous fonction de gestion de transmission d'ordre :

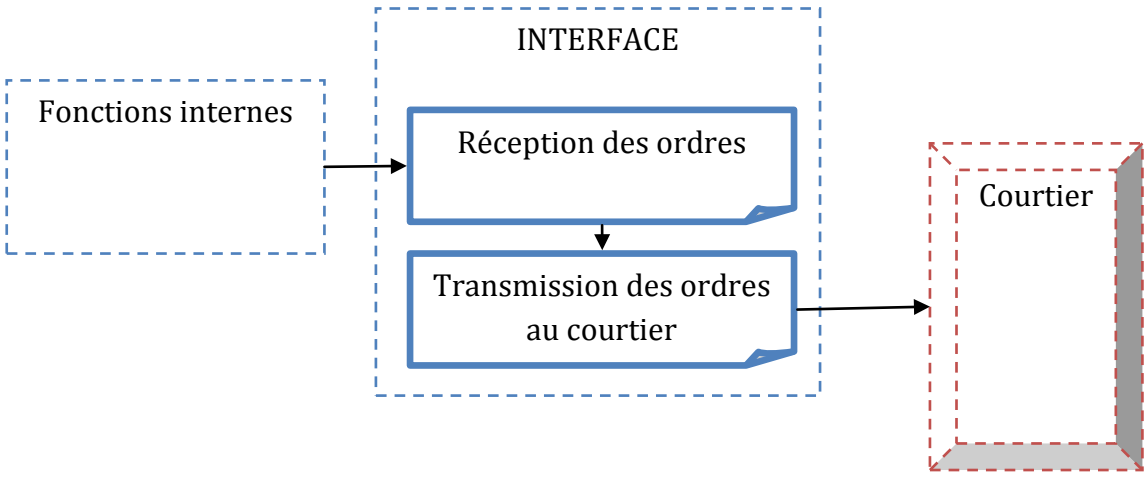


Figure 6 - Schéma de flux de gestion des ordres

3.1.2. FONCTION DE GÉNÉRATION DU SIGNAL

La fonction de génération du signal aura pour but d'analyser les cours de bourse à la recherche d'opportunités d'achat ou de vente. Cette analyse passera par l'exécution et l'évaluation de stratégies. Une stratégie sera un algorithme issu de l'analyse technique, de l'analyse fondamentale ou de l'analyse quantitative. Elle aura pour but d'analyser un cours et de fournir un signal d'achat ou de vente.

La fonction de génération du signal sera donc composée de multiples stratégies qui pourront être utilisées seules ou seront dépendantes les unes des autres afin d'élaborer des stratégies plus complexes.

Chaque stratégie devra être évaluée grâce à :

- La moyenne des profits réalisés.
- La volatilité des profits.
- Le ratio de perte maximale. I. Aldrige (2010) explique que c'est une méthode couramment employée pour évaluer la performance d'une stratégie. Cette méthode se calcule en faisant la différence entre le plus haut point de bénéfice enregistré et le premier plus bas point suivant.

Voici un exemple du fonctionnement de ce ratio d'après:

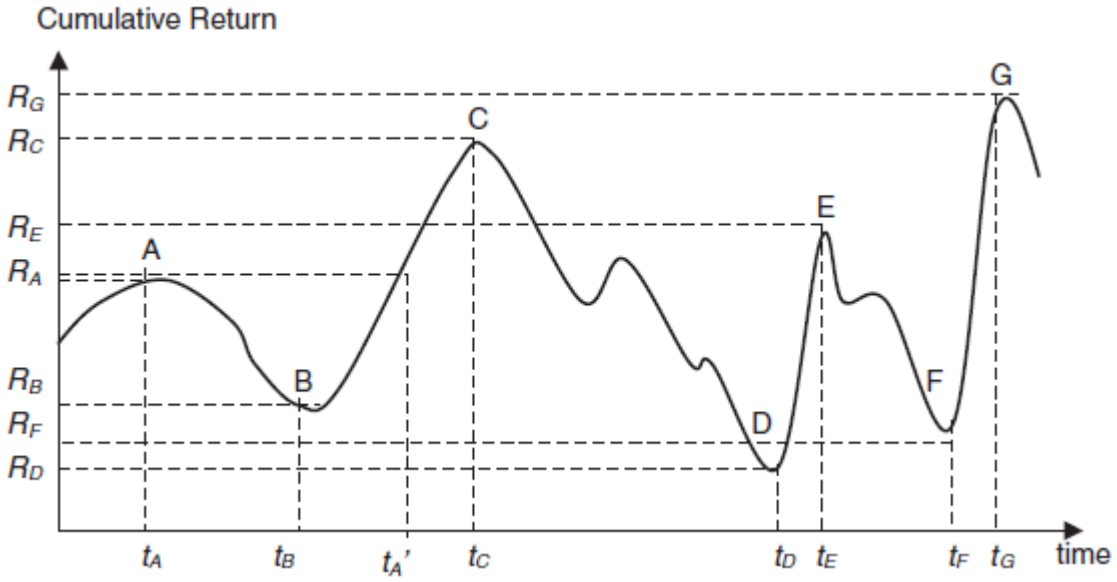


Figure 7 - Calcul de la perte maximale

En première période le plus haut point de bénéfice est le point A. La plus grosse perte est le point B. Le ratio de perte maximale est égale à : (B – A)

Au niveau du point C, le ratio est toujours égal à (B – A) tant que nous n'avons pas atteint un nouveau point de perte maximale inférieur à (B). Une fois le point D atteint, nous avons un plus haut enregistré et un nouveau plus bas enregistré. Donc le ratio de perte maximale devient (D - C)

- Nombre de bonnes décisions. C'est-à-dire quand elles donnent lieu à un profit entre un signal d'achat et de vente (ou l'inverse) :

  Si (cours du signal de vente – cours du signal d'achat > 0) alors r = r+1

  Où ***r*** est le ratio de bonne décision.

- Le ratio temps/profit :

  (tsv – tsa)/Pourcentage (csv – csa)

  Où ***tsv*** est l'heure en milliseconde du signal de vente, ***tsa*** l'heure en milliseconde du signal de d'achat, ***csv*** le cours du signal de vente, ***csa*** cours du signal d'achat.

- Un ratio qualité sur signal d'achat. Calculé pour chaque bonne décision, il indiquera les profits non réalisés entre le moment le plus bas du cours et le signal d'achat.

- Un ratio qualité sur signal de vente. Calculé pour chaque bonne décision, il indiquera les profits non réalisés entre le moment le plus haut du cours et le signal de vente.

Ces indicateurs de performance nous donneront une idée précise de la qualité de la stratégie. Par la suite, nous devrons être en mesure de comparer chaque stratégie avec des ratios bien connus de la finance tel que Sharpe, l'alpha Jensen et le ratio Treynor.

La fonction de génération du signal aura pour fonction à chaque nouveau cours de :

- *Sélectionner* les meilleures stratégies.

  Nous définirons un ensemble de règles s'appuyant sur les critères d'évaluation des stratégies pour sélectionner les meilleures stratégies. Ainsi la fonction de génération du signal fournira à la fonction suivante, fonction d'évaluation, une liste de stratégies à mettre à jour en priorité.

- *Évaluer* les stratégies au fur et à mesure de l'arrivée des cours.

  Les stratégies vont être nombreuses et leurs mises à jour prendront du temps. Le système de trading devant être très réactif, la fonction d'évaluation devra gérer les mises à jour de stratégies par priorité en fonction de la liste fournie par la fonction de sélection. Elle mettra à jour le reste des stratégies en tâche de fonds quand les ressources seront disponibles.

  Au final, la stratégie devra fournir un signal. Le signal doit permettre à la fonction de génération du signal de décider d'une position à transmettre à la bourse. Les signaux peuvent être :

    - L'achat : en prévision que le cours de l'action va monter.
    - La vente : en prévision que le cours de l'action va descendre.
    - Rester à l'achat : pour confirmer une position d'achat.
    - Rester à la vente : pour confirmer une position vendeuse.
    - Aucun signal à transmettre : quand la stratégie n'est pas en mesure de fournir de signal.

La stratégie exécutera ses calculs puis retournera le signal à la fonction de génération su signal.

- *Décider* de la position à prendre.

  Après avoir sélectionné les meilleures stratégies et les avoir mises à jour, nous établirons un ensemble de règles permettant de décider de la position à adopter sur le cours en triant les signaux à l'aide des ratios d'évaluation des stratégies définies précédemment.

A ce stade-là, nous savons quelle est la position à prendre suivant le cours qui vient d'arriver. Les stratégies les plus performantes ont été mises à jour et les moins performantes seront mises à jour en parallèle en basse priorité.

La décision finale est mise à disposition des fonctions suivantes.

3.1.3. FONCTION D'AIDE À L'ANALYSE

La fonction d'aide à l'analyse aura pour objectif d'aider l'analyste au développement de nouvelles stratégies, l'évaluation de celles-ci et de leurs mises en production.

Ces fonctions seront de :

- *Aide au développement de stratégies* via MatLab, R ou Eclipse.

  Cette sous-fonction aura pour but de fournir au développeur une librairie d'accès aux données ainsi qu'un jeu de tests permettant de valider la stratégie techniquement.

- *Évaluation de la stratégie développée***.** La fonction devra fournir un contexte à la stratégie afin de l'alimenter avec les données nécessaires telles que l'historique des cours de bourse.

- *Génération d'un rapport des performances* de la stratégie. Le rapport listera les meilleures performances sur l'historique des cours

3.1.4. FONCTION DU RÉSEAU DE NEURONES

Dans cette partie nous allons étudier et décrire la fonction capable de faire de la reconnaissance de modèle. Pour cela, nous allons utiliser les réseaux de neurones. Nous détaillerons leurs fonctionnements et la façon dont nous allons nous en servir.

Les réseaux de neurones, et, plus généralement l'intelligence artificielle est définie par un de ces fondateurs J. McCarthy comme « la science de faire des machines intelligentes, et plus spécialement des programmes d'ordinateur intelligent. » qui détermine par ailleurs l'intelligence comme « la partie de calcul dans l'habilité à atteindre un objectif. »

Selon S. Russel et P. Norvig (2010), les principaux problèmes pour simuler une intelligence sont :

- *La déduction, raisonnement et résolution de problèmes*. Les humains résolvent la plupart de leurs problèmes en utilisant un jugement intuitif et une déduction étape par étape.

- *La représentation des connaissances*. La représentation des connaissances permet de catégoriser chaque élément qu'un humain a besoin de représenter : des objets, des propriétés, des séries, les relations entre ces objets, des situations, des évènements, des états et le temps.

- *La planification*. La planification sert dans la réalisation d'un objectif et comment régir les étapes pour l'atteindre.

- *L'apprentissage*. C'est la capacité à apprendre d'après différentes données en entrée et de reconnaître des situations, objets, propriétés (…) afin de les classer. Ainsi l'on est capable d'apprendre.

- *L'utilisation du langage naturel*. C'est la capacité de lire et de comprendre le langage humain.

- *Mouvement et manipulation.* C'est la capacité à réaliser des tâches de manipulation d'objets.

- *Perception.* C'est le moyen de déduire des informations via nos sens (vue, ouïe, toucher…)

- *Intelligence sociale.* Ce problème représente la capacité à avoir des émotions et des atouts sociaux.

- *Créativité.* C'est la capacité à imaginer et à avoir une intuition.

- *Intelligence générale.* L'intelligence générale est la capacité à regrouper toutes les capacités citées ci-dessus et pouvoir les utiliser à bon escient.

L'intelligence artificielle en général a développé au cours des décennies de recherche des outils permettant de résoudre ces problèmes. Voici les principaux:

- *Recherche et optimisation.* Beaucoup de problèmes peuvent être résolus en réalisant une recherche intelligente. Par exemple un raisonnement peut-être réduit en réalisant une recherche ou encore la planification utilise la recherche en arbre d'objectif.

- *Logique.* Utilisée pour la représentation de la connaissance et la résolution de problèmes.

- *Méthode probabiliste pour les raisonnements incertains.* Ce domaine permet de combler un manque dans les autres domaines. Il permet de résoudre un problème à partir d'informations incertaines.

- *Méthode d'apprentissage statistique et classeurs.* Les classeurs sont des fonctions qui font de la reconnaissance de modèles pour déterminer la ressemblance la plus proche. Les classeurs peuvent être entraînés à l'aide de méthode d'apprentissage et de statistique.

- *Réseau de neurones.* Les réseaux de neurones sont une représentation artificielle du fonctionnement du cerveau humain. C'est un outil de modélisation de données non linéaire et de décision. Ils sont couramment utilisés pour la recherche de modèle.

- *Théorie du contrôle.* La théorie du contrôle est utile dans le domaine de la robotique.

- *Langages.* Les langages informatiques spécialisés et utilisés pour l'intelligence artificielle en générale sont le Lisp et le Prolog.

Comme mentionné précédemment, la fonction d'intelligence artificielle aura pour objectif de rechercher dans l'historique des cours de bourse, une situation identique à celle qu'elle analyse. Parmi les outils exposés ci-dessus, nous choisirons celui qui est le plus adapté pour réaliser notre objectif : le réseau de neurones.

Le réseau de neurones artificiels est un modèle informatique abstrait du cerveau humain. Le réseau de neurones du cerveau humain est considéré comme la fonction

fondamentale de l'intelligence qui comprend la perception, la cognition et l'apprentissage. Comme pour le cerveau humain, le réseau de neurones artificiels contient des neurones et des interconnexions. Voici le détail de sa composition:

- **Neurone**. Le neurone a pour fonction de choisir si une information doit continuer à transiter sur le réseau. Cette fonction s'appelle la *fonction d'activation*. Nous détaillons le fonctionnement du neurone et ses différentes fonctions plus bas.
- **Synapse**. La synapse a pour fonction de gérer le poids des liens entre les neurones. Pour chaque information qui transite d'un neurone à un autre, la synapse va multiplier cette information par son poids, c'est la fonction *sommatrice.*

Comme le décrit T. Munataka (2008), un neurone a des entrées x*1*, x*2*, …, x*m*. Chaque entrée est multipliée par son poids w*i*, puis transmise au corps du neurone. Les poids représentent la force de la synapse dans un neurone biologique.

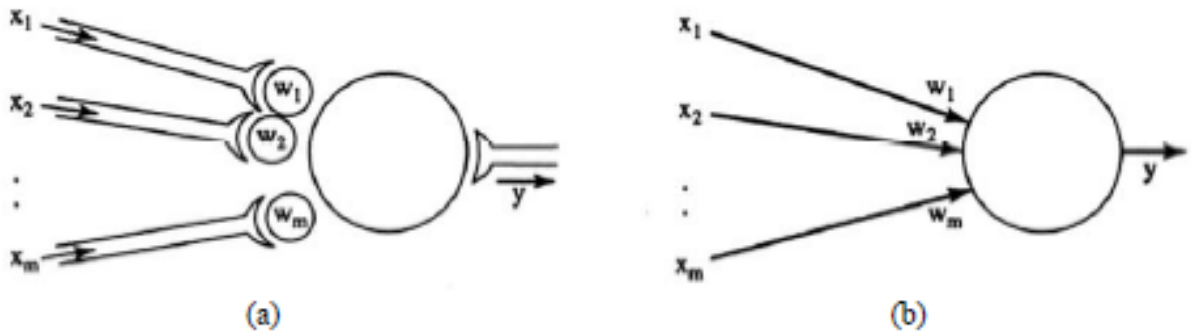


Figure 8 - (a) Modèle d'un neurone biologique (b) Abstraction d'un neurone artificiel

Le neurone calcule x*1*w*1* + x*2*w*2* + … + x*m*w*m* et applique une fonction d'activation qui au final donnera le résultat *y*. Nous pouvons voir le neurone comme une sorte de boite noire recevant un vecteur en entrée d'où il en sortira un résultat scalaire *y*. À noter que, généralement, une application de réseaux de neurones génère aléatoirement le poids de chaque synapse. Ainsi, entre deux exécutions de la même application, on peut avoir deux résultats différents.

Maintenant que nous savons comment fonctionne le réseau de neurones, nous allons devoir lui choisir une typologie, une fonction de sommation et une fonction d'activation. Ces trois paramètres vont avoir un rôle déterminant dans sa performance.

Voici quelques fonctions d'activations :

- **Bipolaire** où les données en entrée sont bipolaires comme vrai ou faux.

- **Compétitive** et utilisée pour forcer une portion de neurones à gagner. Les gagnants sont les neurones qui ont la plus haute sortie.

  Les résultats de chaque neurone sont gardés dans un tableau et passés à cette fonction. La taille du groupe de neurones gagnants est paramétrable. La fonction va d'abord déterminer les gagnants, puis tous les perdants seront positionnés à zéro. Les gagnants vont tous avoir la même valeur laquelle est une division de la somme de toutes les valeurs des neurones gagnants.

- **Gaussien** qui est simplement basé sur une fonction Gaussienne. Elle est très peu utilisée. On s'en sert généralement pour affiner la fonction d'activation.

- **Linéaire**. Cette fonction est utilisée par certains types de réseaux de neurones qui n'ont aucune activation.

- **Log**. Cette fonction s'active à la façon d'une courbe hyperbolique. Elle peut être utile pour prévenir des saturations. Une couche cachée d'un réseau de neurones peut être considérée comme saturée quand les données en sortie sont proches de 1 ou -1.

- **Sigmoïde**. Cette fonction doit être utilisée quand la sortie attendue doit être des nombres positifs. Elle est très fréquemment choisie quand les réseaux sont de type acyclique.

- **Sinus**. Cette fonction est utile seulement quand les données changent périodiquement. Elle peut fournir en sortie des données positives comme négatives.

- **SoftMax**. Essentiellement utilisée dans les réseaux de classification, cette fonction va noter toutes les fonctions en entrée de sorte à ce que leur somme soit égale à 1.

- **TANH** utilise la fonction tangente hyperbolique. Cette fonction est probablement la fonction la plus employée car elle utilise des nombres négatifs et positifs

Aux vues de ces fonctions d'activation, nous utiliserons dans un premier temps la fonction *Sigmoïde* car elle est la seule à nous retourner des nombres positifs (un cours de bourse ne pouvant pas être négatif).

Nous pouvons distinguer trois grandes familles de réseau de neurones :

- **Apprentissage supervisé** :

Un réseau de neurones est dit à apprentissage supervisé lorsque l'on force le réseau à converger vers un état final précis, en même temps qu'on lui présente un modèle.

Cette famille peut se répartir en deux sous-types : avec et sans rétro propagation.

La rétro propagation consiste à corriger le poids des neurones par rapport à la différence entre la sortie du réseau de neurones et la sortie attendue. Ainsi la rétro propagation propage l'erreur commise par un neurone à ses synapses et aux neurones qui y sont reliés. En appliquant cette étape plusieurs fois, l'erreur tend à diminuer et le réseau offre une meilleure prédiction.

- **Sans rétro propagation**

  Exemple : Le perceptron

  Le perceptron est un réseau de neurones à apprentissage supervisé sans rétro propagation. Il est mono-couche et n'a qu'une seule sortie à laquelle toutes les entrées sont connectées. Il fait partie de la famille des classifieurs linéaires.

  Toutefois comme l'ont publié Minsky et Papert (1969) il est démontré que le perceptron était simpliste en terme de capacité de représentation et avait été largement surmédiatisé.

- **Avec rétro propagation**

  Exemple : Le perceptron multicouches

  Le perceptron multicouches est un réseau de neurones à apprentissage supervisé avec rétro propagation. Il est une évolution du perceptron mono couche. Il est organisé en plusieurs couches au sein desquelles une information circule de la couche d'entrée vers la couche de sortie uniquement. Chaque couche est constituée d'un nombre variable de neurones, les neurones de la couche de sorties correspondant toujours aux sorties du système.

- **Apprentissage non supervisés**

Un réseau de neurones est dit à apprentissage non supervisés lorsqu'il est laissé libre de converger vers n'importe quel état final lorsqu'on lui présente un motif.

Exemple : Les réseaux de Kohonen

Les réseaux de Kohonen sont un modèle de neurone plus proche de la réalité. Ces réseaux sont inspirés des observations biologiques du fonctionnement des systèmes nerveux de perception des mammifères.

Comme décrit précédemment, nous souhaitons que la fonction d'intelligence artificielle soit en mesure de faire de la reconnaissance de modèle. Pour cela elle doit s'appuyer sur un historique de données. Nous allons donc alimenter cette fonction de sorte qu'elle apprenne à partir de cet historique. Nous serons dans un mode d'apprentissage supervisé. Et nous utiliserons la fonction « évoluée » de ce réseau, le mode multicouches.

Le nombre de neurones utilisés dans la couche d'entrée et la couche de sortie sont habituellement déterminés par la nature du problème à traiter. Comme le donne en exemple T. Munakata (2008), pour un problème de reconnaissance de caractères, chaque caractère est découpé en une grille de 100 points. Donc le nombre de neurones dans la couche d'entrée sera de 100. Pour la couche cachée, il n'y a pas de méthode pour définir le nombre de neurones à utiliser. Nous définirons le nombre de neurones nécessaires en réalisant des tests et en comparant le taux d'erreur obtenu.

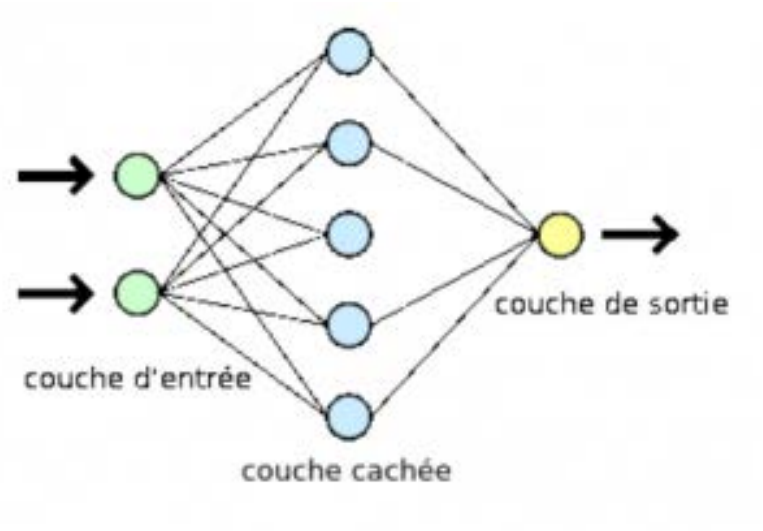


Figure 9 - Schéma d'un réseau de neurones artificiels multicouches

Enfin à ces familles de réseaux de neurones, nous pouvons ajouter deux spécificités sur la manière dont sont traitées les données à l'intérieur du réseau :

- Acycliques (feed-forward en anglais)

Ce type de réseau de neurones est le plus simple. Dans ce type de réseau, l'information va dans une seule direction : de l'entrée à la sortie.

- Cycliques (récurrents)

  A l'opposé du réseau acyclique, ce type de réseau réalimente ses propres entrées avec l'information en sortie. Cela signifie que les niveaux d'activation forment un système dynamique qui peut atteindre un état stable ou présenter des oscillations, ou même un comportement chaotique. Il est idéalement utilisé pour simuler la mémoire à court terme (S. Russel, P. Norvig 2010)

Souhaitant simuler un comportement par de la reconnaissance, nous nous en tiendrons au type simple, le type acyclique.

Concernant la fonction de sommation, nous pouvons distinguer trois fonctions principales:

- **Les synapses pondérées**. Cette fonction connecte chaque neurone avec un neurone de la couche suivante. D'après la documentation du réseau de neurones Encog (2011), il est généralement utilisé lorsqu'un réseau nécessite un apprentissage.

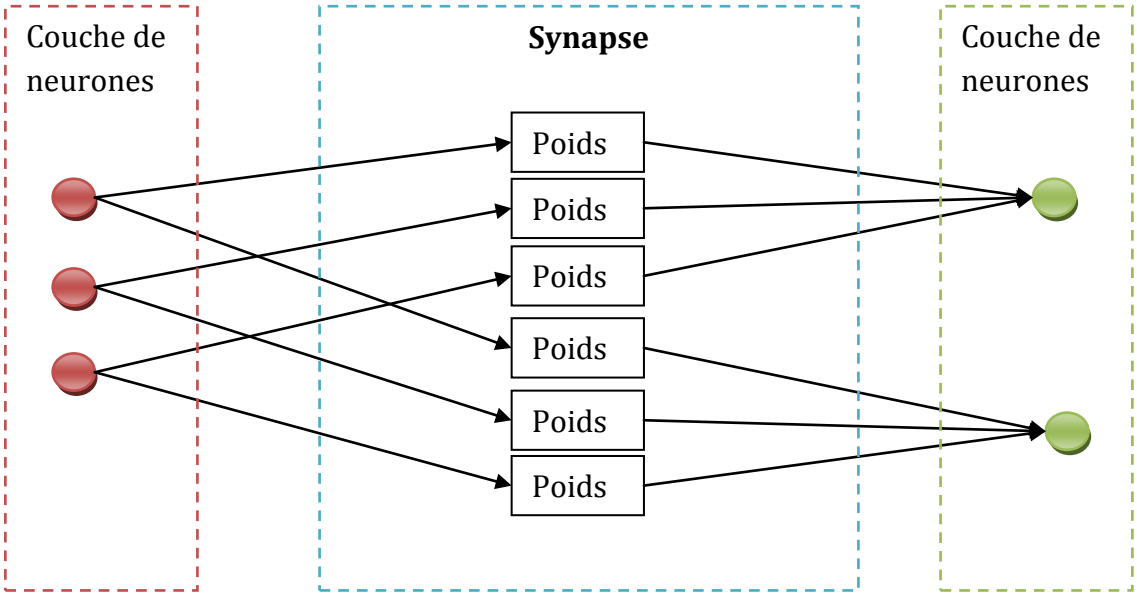


Figure 10 - Schéma représentant le fonctionnement des synapses pondérées

- **Les synapses sans poids.** Cette fonction n'ajoute pas de poids à la synapse. Il est donc impossible pour un réseau d'apprendre. L'information transitera à travers

la synapse de neurone en neurone sans être modifiée (à part par le neurone lui-même)

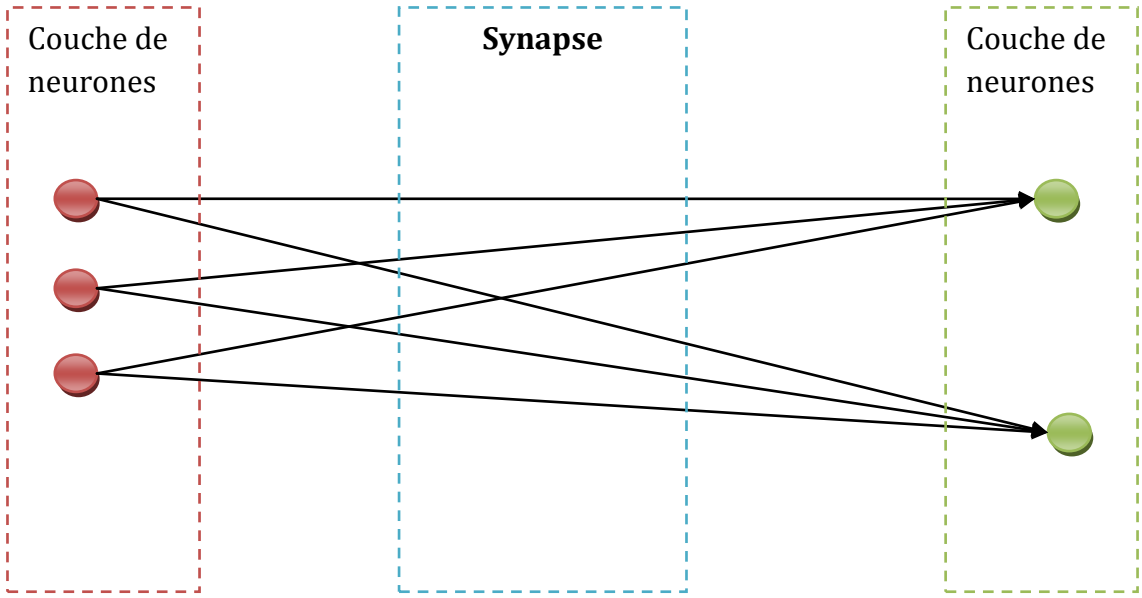


Figure 11 - Schéma de fonctionnement des synapses sans poids

- **Les synapses une à une.** Cette fonction procède comme la synapse sans poids. Elle ne gère pas de poids, mais elle connecte les neurones de la couche précédente au neurone de la couche suivante un à un.

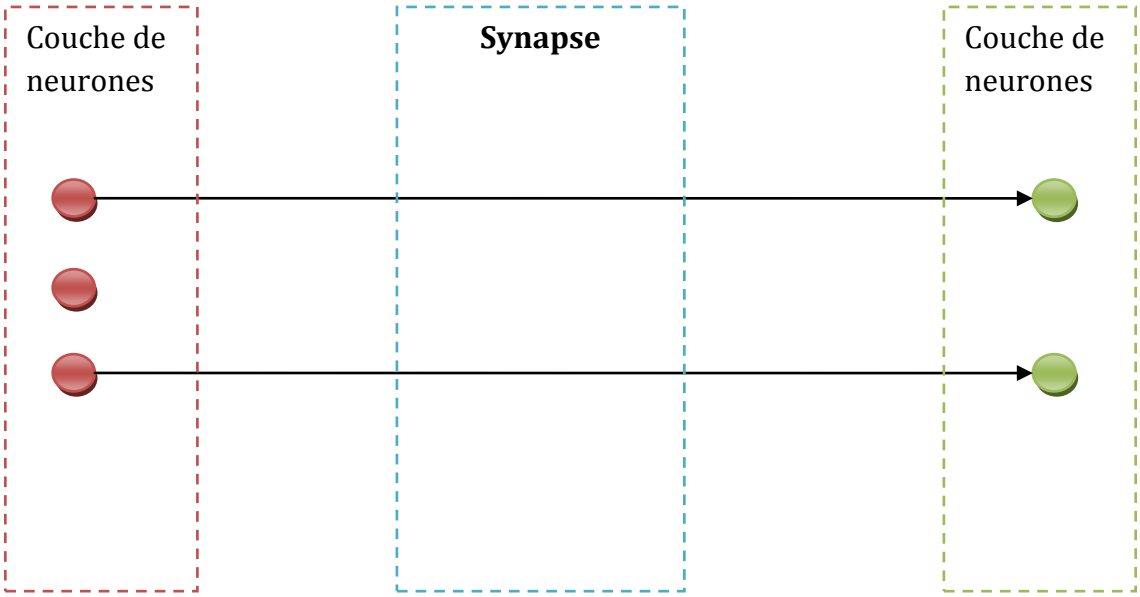


Figure 12 - Schéma de fonctionnement des synapses une à une

- **Les synapses directes.** Les synapses directes sont utiles lorsque l'on souhaite envoyer une copie complète des résultats d'une source de neurones à un neurone de destination.

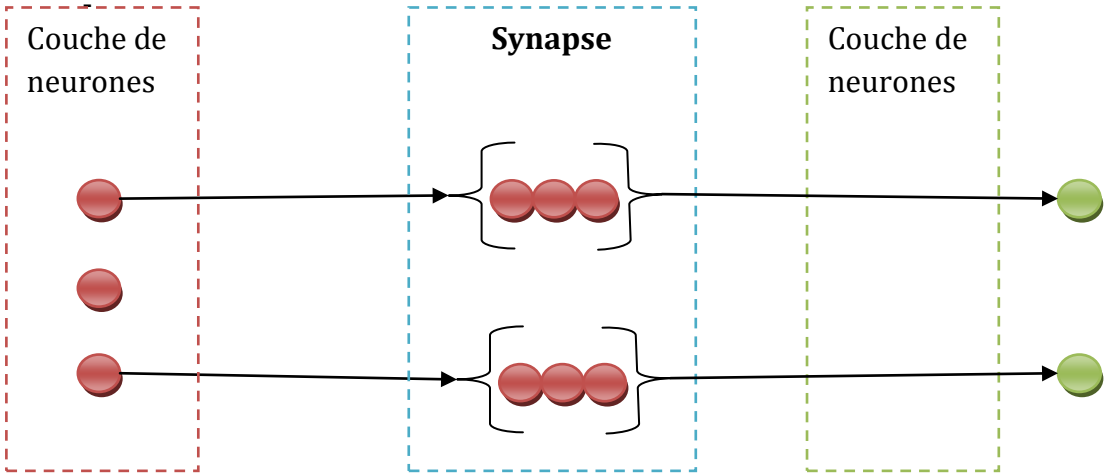


Figure 13 - Schéma de fonctionnement des synapses directes

Taux d'erreur

Afin de connaître le niveau de fiabilité des données générées par le réseau de neurones, nous définirons un taux d'erreur. Lorsqu'un nouveau cours sera soumis au réseau de

neurones, celui-ci va rechercher dans l'historique un cours qui se rapproche au plus près de celui fourni en entrée. Pour cela, il va calculer la *distance* entre deux modèles. Ainsi cette distance sera donnée sous forme d'un taux d'erreur.

Nous avons décrit ici le fonctionnement de l'intelligence artificielle et plus particulièrement celui d'un réseau de neurones. Nous en avons déterminé les différents types et défini les paramètres que nous utiliserons.

Ce réseau de neurones agissant comme une boite noire d'un point de vue extérieur (impossible de connaître l'état des neurones une fois paramétré), les choix des paramètres ou la typologie du réseau de neurones évolueront au fur et à mesure du temps en fonction des retours et des taux erreurs que nous aurons.

3.1.5. FONCTION DE GESTION DU RISQUE

La fonction de gestion du risque devra être en mesure d'estimer le risque encouru si l'on venait à prendre position. Position qui serait prise après étude des signaux fournis par la fonction de génération du signal et de la fonction de réseau de neurones.

Par rapport à ces deux signaux et le cours actuel sur lequel ces signaux portent, cette fonction utilisera un indicateur de calcul de risque d'exposition. Cet indicateur, le *Value At Risk – VaR* (Valeur sous risque) est parmi le plus utilisé en finance. Il permet de faire une estimation des pertes qui ne devraient pas être dépassées.

Parmi les différentes méthodes de calcul de l'indicateur VaR (méthode historique, variance-covariance et Monte-Carlo) nous utiliserons la méthode variance-covariance car elle fournit le meilleur rapport efficacité/temps de calcul (la méthode historique ne donnant pas un niveau de risque assez élevé et la méthode de Monte-Carlo demandant trop de puissance de calcul).

Voici la formule de variance-covariance : **profit attendu x σ x $\sqrt{T}$**

Où :

profit attendu est le profit que nous espérons faire.

T est la période sur laquelle nous sommes exposés.

σ est l'écart-type.

Ainsi cette formule nous donnera le risque encouru en pourcentage. Par rapport à ce résultat, nous pourrons définir la quantité de sous-jacents (devises, options, obligations…) que nous souhaiterions acheter. C'est-à-dire que plus le risque sera élevé, moins nous achèterons.

3.1.6. FONCTION DE MONITORING

La fonction de monitoring permettra d'obtenir des informations sur l'état du système (d'un point de vue technique) et sur l'état des investissements et des performances en cours. Nous regrouperons dans cette fonction, la fonction d'administration qui sera chargée de rassembler les paramètres du système technique (serveur, port, identifiants…) et du système applicatif (activation des stratégies, seuil de pertes…)

Détaillons ces fonctions:

- ***La fonction de monitoring***. Elle aura pour rôle de nous fournir des rapports et des détails sur l'état du système.

  Par fonction, elle nous rapportera pour :

  - L'interface, le nombre de messages à la seconde et les alertes (message non transmis, message malformé, exception de l'application)
  - La génération du signal : Les stratégies exécutées, le rapport de performance de chaque stratégie et les décisions prises par rapport à quelles stratégies.
  - Le réseau de neurones : Les temps d'exécution, les taux d'erreur pour chaque décision et leurs prévisions.
  - La gestion du risque : Les décisions prises, les taux de risque et les quantités évaluées en fonction du risque.

  L'aide à l'analyse ne sera pas monitorée étant une fonction qui fournit un rapport à la fin de son exécution.

- ***L'administration.*** Elle nous permettra de paramétrer les différentes fonctions du système ainsi que ses paramètres.

Également par fonction, elle nous permettra pour :

- L'interface : Configurer les intervenants extérieurs (courtier) et de configurer le système d'envoi de message
- La génération du signal : Activer/Désactiver les stratégies et de charger une nouvelle stratégie
- Le réseau de neurones : Activer/Désactiver un réseau de neurones
- La gestion du risque : Définir le seuil de perte acceptée

Nous avons défini dans ce chapitre les fonctions de notre système de trading. Nous allons maintenant le spécifier techniquement.

3.2. SPÉCIFICATIONS TECHNIQUES

Grâce au chapitre précédent, nous avons une vision des fonctions de notre système dans sa globalité. Nous allons pourvoir développer le système techniquement. Nous ne parlerons plus de fonction, mais de modules.

3.2.1. LE MODULE D'INTERFACE

L'interface est la fonction qui sera en mesure de récupérer les cours de bourse et transmettre des ordres avec des systèmes externes (courtiers, institutions financières). Pour cela, elle devra être capable de gérer une communication externe (flux entrant : cours de bourse, flux sortant : transmission d'ordre) et un dialogue interne (flux sortant : cours de bourse, flux entrant : ordres). Elle devra donc être en mesure d'utiliser différents protocoles (le langage interne et le format de données au système de trading étant différent du courtier) et différents moyens de communication. Les systèmes externes étant amenés à changer et les fonctions du système de trading à évoluer, il faut que notre interface puisse être indépendante de tout changement. Elle doit pouvoir

adopter un nouveau protocole de communication sans changer sa structure interne. Elle devra donc être interopérable.

Les courtiers/institutions financières permettent à leurs clients de s'abonner à un flux de cours de bourse où ils peuvent recevoir les nouveaux cours établis, les carnets d'ordres… Ils sont aussi en mesure de prendre de ces clients des ordres qu'ils vont « exécuter » (Le but réel est de trouver une contrepartie qui cherche à acheter ou vendre votre sous-jacent).

Pour dialoguer avec les acteurs externes, nous devons définir un protocole et un moyen d'utiliser ce protocole pour transformer les messages reçus dans un format standard à notre système. D'après le site de l'organisation FIXProtocol(2012) qui gère le protocole d'échange financier « FIX » pour « Financial Information Exchange », 90% des transactions financières se font via leur protocole. La spécification du protocole FIX est issue d'un regroupement d'institutions financières notamment les grandes places boursières mondiales. Les spécifications complètes de ce protocole se trouvent sur ce même site.

Pour gérer ce protocole, nous utiliserons un gestionnaire (« engine » en anglais) pour le manipuler et ainsi communiquer avec les systèmes externes. Ce gestionnaire sera en mesure de recevoir les cours de bourse et de nous les transmettre dans un format qui nous permettra de les manipuler. Ainsi, parmi tous les gestionnaires de protocole FIX disponibles sur le marché, nous opterons pour une solution gratuite et de préférence open-source. Le gestionnaire « *QuickFIX* » est un gestionnaire populaire, simple d'utilisation et répondant à nos besoins de performance.

Nous noterons toutefois que certains courtiers ou institutions financières peuvent mettre à disposition des librairies permettant de dialoguer avec leurs systèmes.

Voici un diagramme de flux de l'interface avec un système externe :

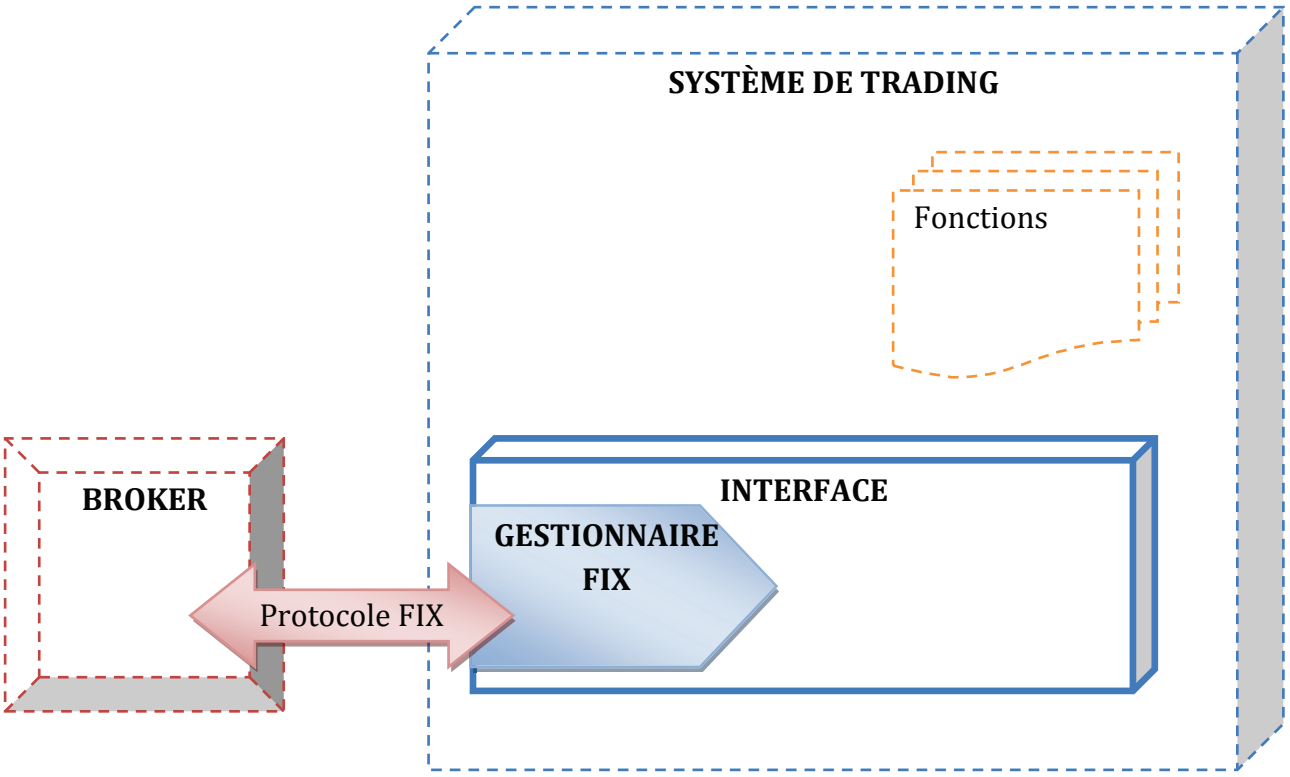


Figure 14 - Diagramme de flux externe de l'interface

Le nombre de messages (cours de bourse, …) échangés avec le système externe est conséquent. Selon le site web de NYSE Euronext (2012) la fréquence de réception des messages est de l'ordre de 800 000 par seconde sachant qu'un message pèse environ 200Ko donc un total d'environ 156Mo à la seconde. Il faut donc que l'interface puisse faire face à une telle charge.

En interne, notre interface gérera le dialogue avec les différents modules. Ainsi, lorsqu'elle recevra un cours de bourse, elle le mettra à disposition du système tout entier. Les fonctions intéressées s'abonneront à cette catégorie de message. Lors de la mise à disposition d'un nouveau message, ils seront automatiquement avertis.

Cette typologie de message/évènement asynchrone est détaillée par G. Hohpe et B. Wolf (2003). Ils répondent à cette problématique en présentant un modèle d'intégration d'entreprise dénommé « Messaging » (« Messagerie » en français).

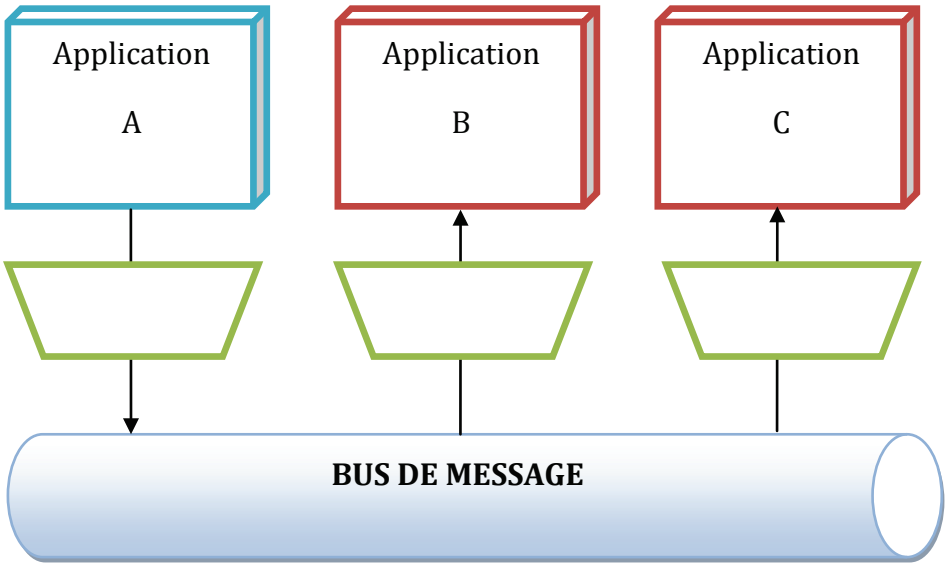


Figure 15 - Modèle d'intégration d'entreprise "Messagerie"

L'application A envoie un message via un bus qui va être en charge de le distribuer aux applications abonné. Le message transitant dans le bus sera routé vers l'application B et C. Puis le message sera supprimé du bus.

Dans notre cas de figure, l'interface est représentée par l'application A. Les autres fonctions de notre système sont représentées par les applications B et C.

Le diagramme suivant présente cette typologie appliquée à notre système :

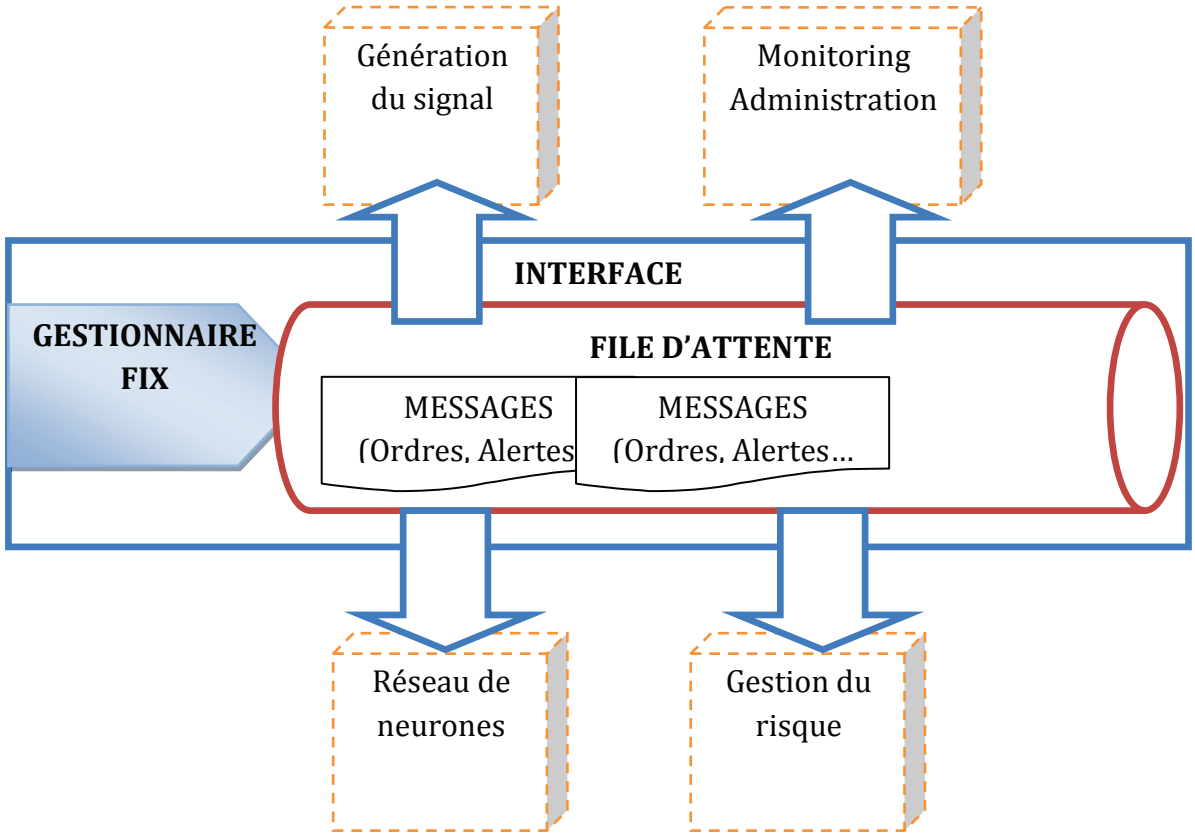


Figure 16 - Diagramme technique de l'interface en interne

L'arrivée des cours de bourse se fait via le gestionnaire du protocole FIX. L'interface se charge de la conversion du message dans un format standard et le poste dans le bus, ici schématisé par la file d'attente. Nos autres fonctions, précédemment abonnées au bus, reçoivent le message. Chaque fonction détaillera dans le chapitre qui lui est attribué le type de message dont elle souhaite être avertie.

L'avantage d'un tel type d'architecture est que les différentes fonctions peuvent être exécutées sur des machines différentes. Pour l'interface qui nécessite de fortes ressources mémoire pour gérer le volume de messages, c'est un avantage que de pouvoir s'exécuter sur une machine distincte du reste du système afin d'assurer sa fonction.

3.2.2. LE MODULE DE GÉNÉRATION DU SIGNAL

Le module de génération a pour but l'exécution des stratégies et la prise décision du signal final. Il sera donc composé de plusieurs stratégies et d'une fonction de décisionnel.

Voici un graphique qui illustre les flux de cette fonction :

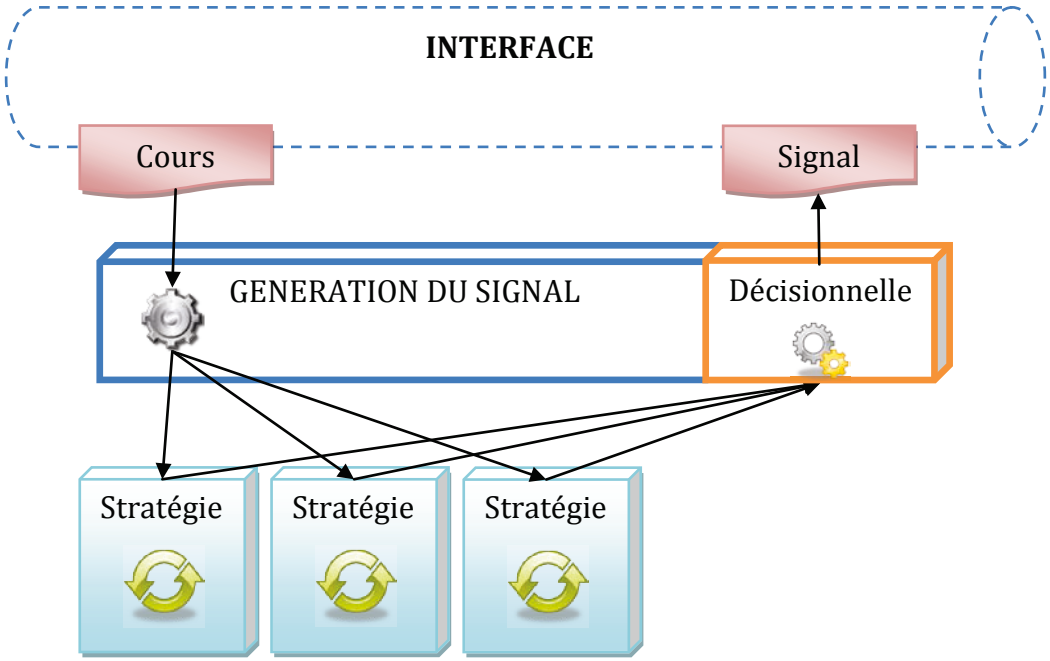


Figure 17 - Diagramme des flux de la génération d'un signal

Détaillons ce flux:

1. Le module de génération du signal reçoit un nouveau cours via le système de bus de l'interface.
2. Il charge les stratégies, règle les dépendances entres les stratégies, puis les trie en fonction de leurs performances antérieures.
3. Il charge les données nécessaires à leurs exécutions.
4. Il exécute les stratégies une à une puis récupère le signal de chacune.
5. Suivant les règles décisionnelles fournies, il décide du signal final.

6. Enfin il poste le signal sur le bus de l'interface pour que les modules intéressés puissent en prendre connaissance.

Au préalable de ces exécutions, il faudra que le module s'abonne au cours de bourse, transitant sur le bus de l'interface, afin d'être averti de tout nouveau cours. Le nouveau cours de bourse sera transmis aux stratégies afin d'être analysé. Une stratégie est un algorithme qui déduira, à partir de ce cours et de l'historique des données, des opportunités d'achat et de vente. De nombreuses stratégies existent déjà dans le domaine de la bourse. Ces stratégies sont des indicateurs techniques tels que le MACD (Moving Average Convergence Divergence ou convergence et divergence des moyennes mobiles), ou MME (Moyenne mobile exponentielle)… Une même stratégie pourra fonctionner avec différents paramètres, comme le MACD peut prendre en paramètres différent cours (devises, options…) sur différents espaces de temps (minute, jour, mois…) et d'autres paramètres propres à lui (période rapide, période lente, signal). Ils permettront de détecter un début de comportement et pourront ainsi prédire la suite du comportement. À ce moment ils donneront un signal d'achat s'ils prédisent que la suite du comportement va faire monter le cours de bourse (ou un signal de vente dans le cas contraire).

Le module de génération du signal tient à jour des statistiques sur les performances des stratégies qu'ils exécutent afin de savoir à quel point il peut se fier aux signaux des stratégies. Il fera donc une somme des performances des signaux à l'achat, et la somme des performances des signaux à la vente. Plus l'écart sera grand entre ces deux sommes, plus la décision finale sera fiable (un grand écart signifiant une forte fiabilité).

Exemple :

10 stratégies donnent un signal d'achat. Le module récupère la performance de chacune de ces stratégies et l'additionne, résultat : 120.

13 stratégies donnent un signal de vente. Le module récupère la performance de chacune de ces stratégies et l'additionne, résultat : 52.

La figure suivante illustre cet exemple :

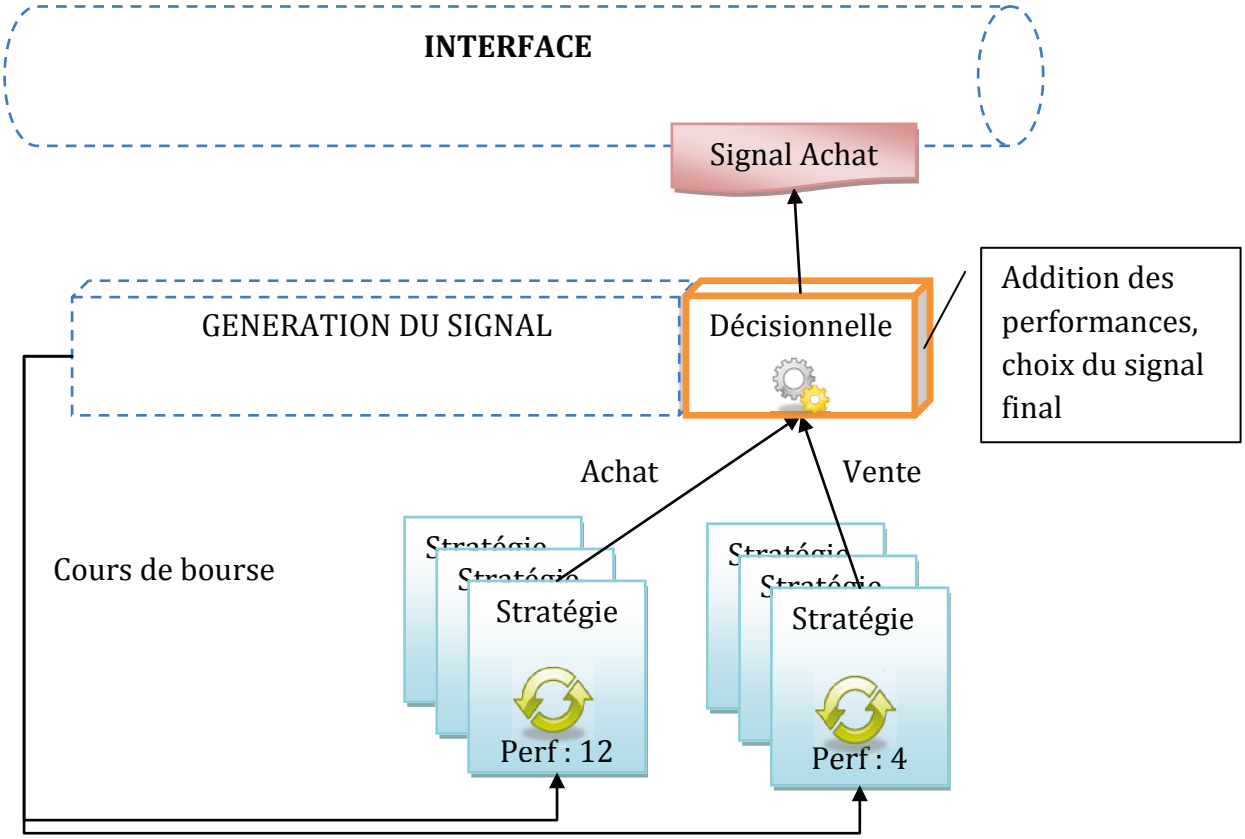


Figure 18 - Schéma de fonctionnement de la fonction décisionnelle

Dans ce cas, même si les stratégies fournissant un signal de vente sont plus nombreuses, leurs performances sont bien moins fiables. L'écart avec la somme des performances des stratégies qui fournissent le signal d'achat permet au module de génération du signal de prendre la décision d'achat.

Concernant les données nécessaires aux stratégies, le module devra être en mesure de fournir l'historique des cours de bourse. Il devra avoir chargé les données pour le sous-jacent sur laquelle porte l'analyse de la stratégie. D'autre part, l'analyse d'une stratégie peut dépendre des résultats d'une autre stratégie. Au chargement de la stratégie et lors de l'arrivée d'un nouveau cours, le module de génération du signal devra être capable de régler les dépendances entre les stratégies. Pour chaque stratégie, il leur demandera leurs dépendances. Ainsi une stratégie s'exécutant avec une forte priorité (du fait de ces performances) pourra dépendre d'une stratégie ayant une faible priorité. De ce fait le module devra exécuter, malgré tout, la stratégie de faible priorité avant les autres.

3.2.3. LE MODULE D'AIDE À L'ANALYSE

L'aide à l'analyse permettra le développement de nouvelles stratégies. Voyons les étapes de développement d'une nouvelle stratégie :

1. Développement de la stratégie

   L'analyste développeur aura à disposition un environnement de développement et une librairie d'accès aux données. Il pourra baser sa stratégie sur de simples formules mathématiques ou s'appuyer sur les résultats d'autres stratégies.

   Pour chaque stratégie, l'analyste développeur choisira les données qui seront exécutées sur sa stratégie. Il pourra choisir de l'exécuter sur une seule ou plusieurs actions et sur une partie ou l'ensemble des cours.

2. Chargement de la stratégie dans le système d'analyse

   Lors du chargement de la stratégie, une copie en sera sauvegardée afin de garder un historique. Pour chaque stratégie, l'ensemble des paramètres qui lui sont associés sera aussi sauvegardé.

3. Exécution de la stratégie

   Quand une stratégie est exécutée, les paramètres et les données lui sont fournis au fur et à mesure. Une fois exécuté, ses résultats sont sauvegardés. Ces résultats sont les différentes décisions d'achat/vente qu'elle aura déduit au fur et à mesure de son exécution.

4. Génération d'un rapport des performances de la stratégie

   La génération du rapport interviendra une fois tous les paramètres passés sur la stratégie. Il fournira les détails sur l'exécution en précisant la stratégie qui a fourni le meilleur résultat en fonction des paramètres.

Voici le schéma avec les étapes que nous venons de décrire :

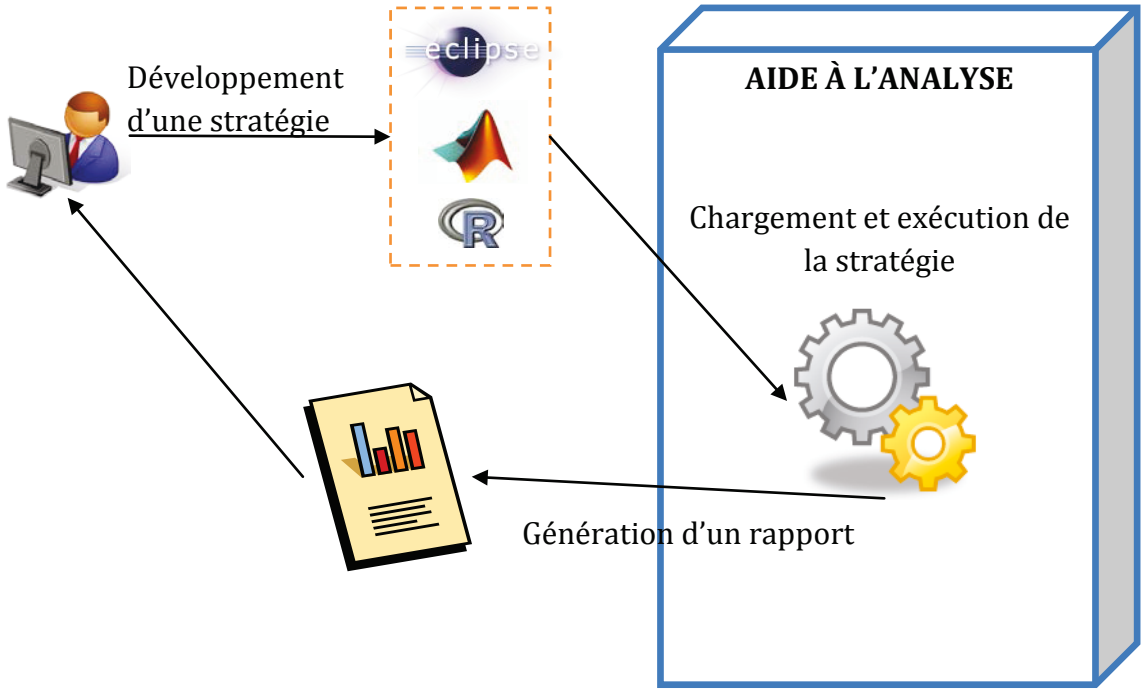


Figure 19 - Schéma de développement d'une stratégie

Nous allons détailler chacune de ces parties.

Développement d'une stratégie

Lors du développement d'une nouvelle stratégie, le développeur aura donc à disposition un environnement de développement (eclipse, matlab, R) qui lui permettra de coder la stratégie dans le langage propre à l'environnement de développement. À cet environnement de développement s'ajoutera une librairie de fonction. Cette librairie de fonction lui permettra :

- D'accéder aux données historiques des cours de bourse
- D'avoir des fonctions mathématiques (calcul, analyse de graphe…)
- D'accéder aux anciennes stratégies

Cette dernière fonction « accès aux anciennes stratégies » pourra se faire dans un premier temps en prenant connaissance via une interface web des performances globales des stratégies. Ainsi le développeur pourra choisir s'il souhaite faire dépendre sa stratégie d'une autre stratégie.

Voici le schéma de l'environnement du développeur :

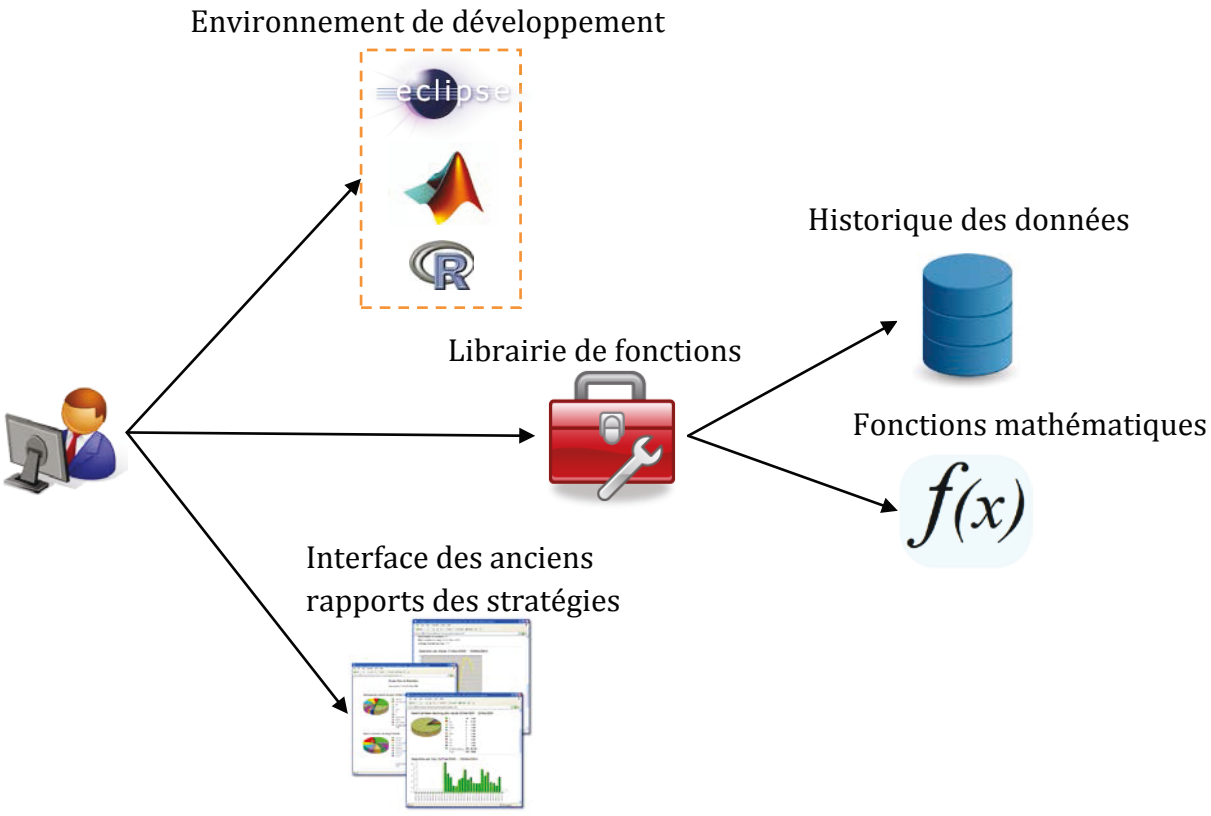


Figure 20 - Environnement de travail d'un développeur

Le chargement et l'exécution d'une stratégie se feront directement par l'environnement de développement. Ainsi, aucune connaissance technique supplémentaire ne sera nécessaire au développeur.

Lorsque le développeur développe une stratégie, il spécifie, la plage de temps sur laquelle il veut que sa stratégie soit exécutée, l'écart de temps (seconde, minute, heure…) et le type de sous-jacents. Ainsi, lors de l'exécution de la stratégie, le module d'analyse récupérera en premier les paramètres de la stratégie. Il chargera les sous-jacents depuis la base de données (Devises, actions, obligations…), puis préparera l'environnement d'exécution de la stratégie. Ensuite, il sauvegardera une copie de la stratégie au cas où un autre développeur souhaiterait l'utiliser ultérieurement. Il sauvegardera les paramètres, puis débutera l'exécution même de la stratégie. Pour chaque signal que fournira la stratégie, le module d'aide à l'analyse sauvegardera le signal (Achat/Vente), le prix, le sous-jacent (ex: devise EUR/USD, action BNP, …), l'écart de temps (ex : minute, jour, mois…). Puis il associera cette sauvegarde à la stratégie et aux paramètres. Ainsi le

module d'aide à l'analyse aura une copie entière de la stratégie et sera capable de la ré-exécuter à tout moment ou de la mettre à jour.

Une fois exécutée et les résultats sauvegardés, le module d'aide à l'analyse générera un rapport au format PDF avec les différentes données nécessaires pour évaluer la stratégie. Le développeur pourra à ce moment-là, savoir comment optimiser sa stratégie ou la soumettre à ses responsables pour la mise en production.

Ce rapport contiendra des données concernant l'exécution globale de la stratégie (temps d'exécution, nombre d'exécutions par rapport au paramètre), le détail concernant le meilleur résultat obtenu (le graphe et avec quel broker cela a été le plus profitable, et le pourcentage total), puis les autres résultats classés par performance.

Voici un exemple de rapport :

Stratégie MACD

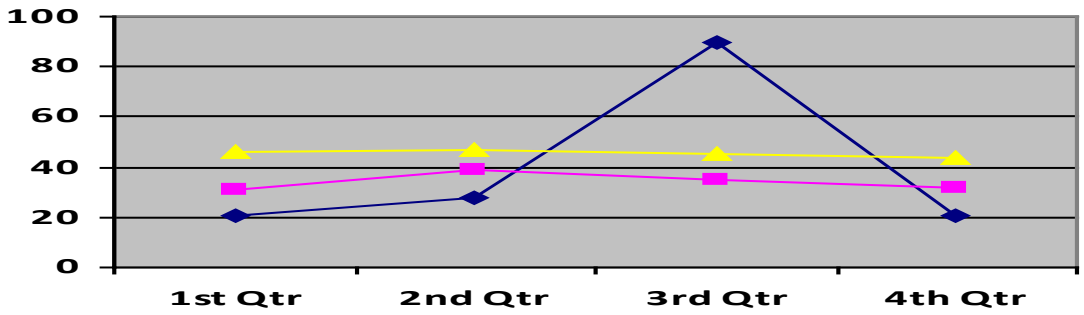
Nombre d'exécutions : 2312
Temps total d'exécution : 2j 18h 23m

Meilleur résultat

Paramètres : Fast 8 – Slow 12 – Signal 9	-	Pourcentage : 18%

Sous-jacents : DEVISE EUR/USD
Type de prix : CLOSE
Écart de temps : Jours
Date : du 01/01/2012 au 01/02/2012

Nombre total d'achat/vente : 652
Achat : 312
Vente : 340



Temps d'exécution : 12 min
Broker : FXCM (2.6pips)

---

100 autres meilleurs résultats

Paramètres	-Sous-jacents -	Type de devise -	Écart de temps –	Date –	Pourcentage
26, 12,9	DEVISE	EUR/USD	Jour	12/02/2012	14%

Figure 21 - Exemple de rapport généré par le module d'aide à l'analyse

Comme vu précédemment, le développeur pourra aussi prendre connaissance des performances des stratégies déjà exécutées. Pour cela, il aura accès à une simple application web qui listera les stratégies, leurs paramètres, et les performances de celles-ci.

3.2.4. LE MODULE DE RÉSEAU DE NEURONES

Le module de réseau de neurones va être le module le plus complexe à mettre en place. Il requiert de bien maîtriser le sujet des réseaux de neurones et notamment les aspects techniques des réseaux de neurones. Toutefois, les systèmes de prédiction de marché étant très peu documentés en général, nous mentionnerons toutes décisions techniques qui pourraient être purement arbitraires.

Comme décrit dans les spécifications fonctionnelles, ce module sera doté d'un réseau de neurones. Le type de ce réseau de neurones sera à apprentissage supervisé et à rétro propagation.

Selon J.A. Freeman et D.M. Skapura (1991), les réseaux à rétro propagation sont très utiles pour résoudre des problèmes de reconnaissance de modèles complexes. Ce type de réseau qu'illustre la figure suivante, est conçu pour être multicouche et à apprentissage supervisé.

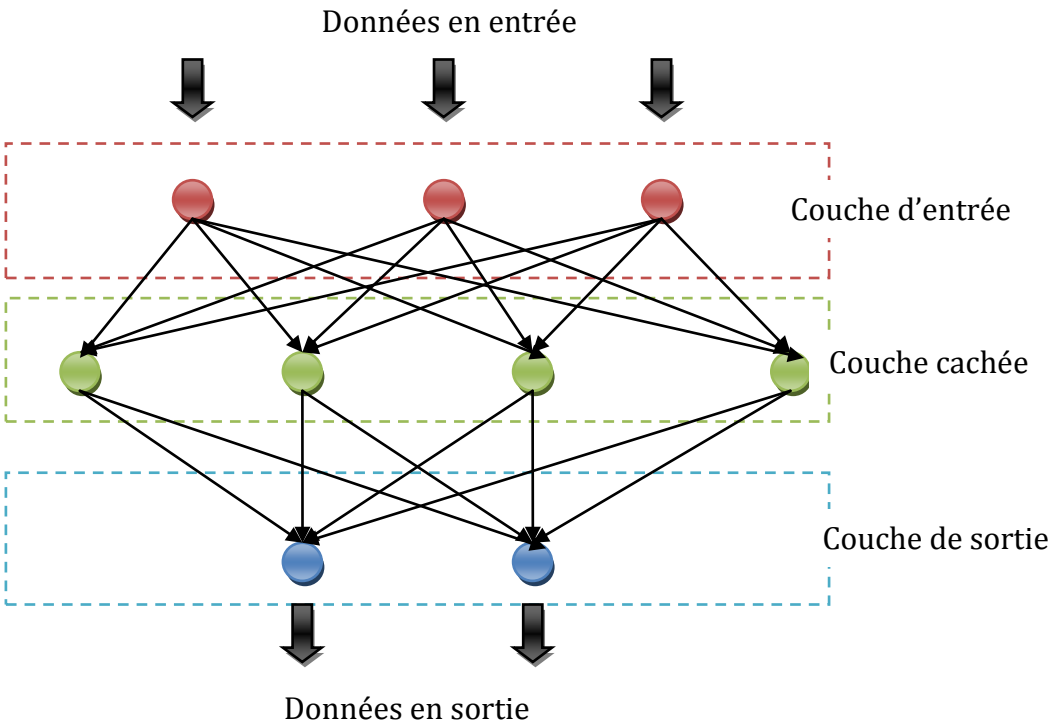


Figure 22 - Schéma d'un réseau multi couches à rétro propagation

Ce réseau de neurones qui doit faire de la reconnaissance de modèles va devoir faire face à deux problèmes concernant le cours de bourse :

- Reconnaître le cours de bourse historique qui se rapproche le plus du cours actuel.
- Générer une suite probable.

Voici un simple cours de bourse représentant une baisse suivi d'une hausse de cours illustré par la figure suivante :

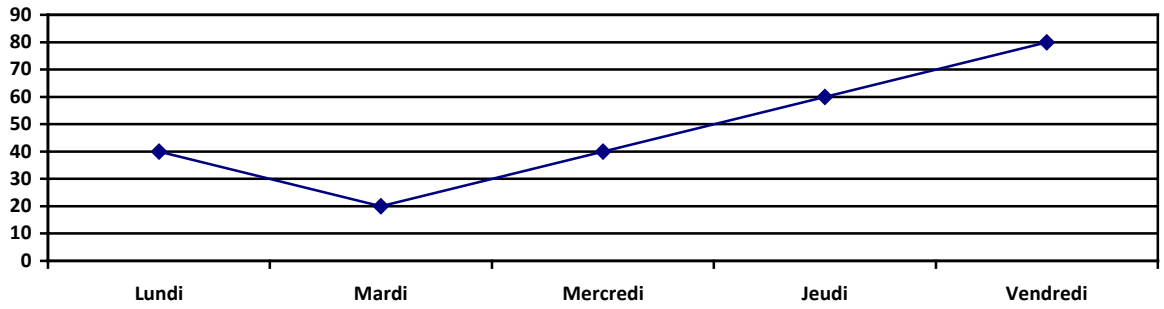


Figure 23 - Cours de bourse avec une hausse linéaire

Lundi l'action vaut 40, mardi elle vaut 20, mercredi elle vaut 40, jeudi elle vaut 60 et enfin vendredi elle vaut 80. Si nous mettons une table de correspondance comme pour la reconnaissance de caractère de type :

Lundi=40, Mardi=20, Mercredi=40, Jeudi=60, Vendredi=80

Il sera donc possible à partir de ce tableau de correspondance de retrouver dans l'historique des cours de bourse le modèle correspondant à ce cours de bourse.

La semaine suivante, le cours de bourse est représenté par la figure suivante :

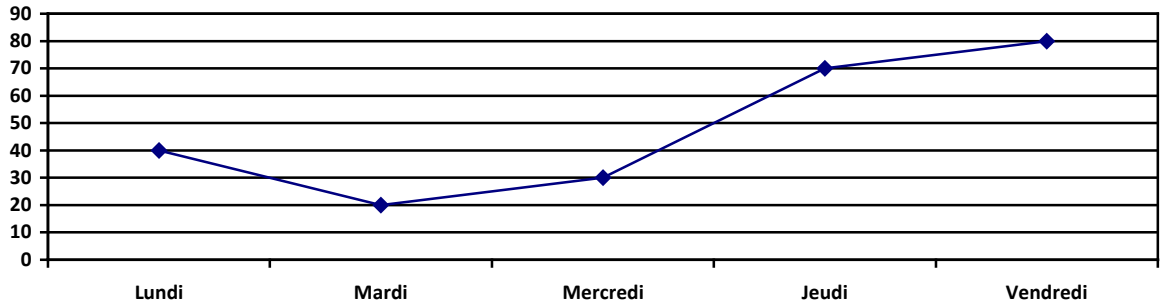


Figure 24 - Modèle de cours de bourse avec une hausse non linéaire

Nous pouvons constater que lundi, mardi et vendredi, les cours sont identiques au graphique précédent. Cependant mercredi et jeudi le cours a dévié et n'est plus linéaire comme précédemment. Nous sommes tout à fait d'accord pour dire que la hausse du cours à bien lieu après la baisse et que le résultat reste le même car le vendredi l'action

atteint le prix de 80. Cependant, le modèle n'est pas reconnu dans notre table de correspondance.

Le caractère modifié dans notre premier exemple et la déviation du cours de bourse empêche la reconnaissance de modèle. Ces modifications « naturelles » sont appelées « bruit ».

Le problème que le bruit induit sur l'image ou sur le cours de bourse est difficile à résoudre par un ordinateur dû à l'incompatibilité entre la logique de la machine et le problème. Le réseau de neurones à rétro propagation permet de palier le problème complexe de reconnaissance de modèles. Nous allons maintenant décrire son fonctionnement.

Le réseau de neurones à rétro propagation va dans un premier temps *apprendre*. Pour ce faire, il lui faut un jeu de données que nous souhaitons faire analyser. Ce jeu de données va d'abord être traité par la couche d'entrée du réseau. Ensuite ces données seront propagées aux couches intermédiaires qui sont les couches cachées jusqu'à ce qu'elles produisent un résultat. Ce résultat est ensuite comparé au modèle que l'on souhaite obtenir. Le réseau calcule la différence entre le modèle et le résultat qu'il a généré. Pour chaque donnée du résultat, il en résulte un signal d'erreur. Ces signaux sont ensuite re-propagés via le réseau depuis la dernière couche cachée jusqu'à la première couche cachée. Chaque neurone des couches intermédiaires reçoit le signal d'erreur de la donnée à laquelle il a contribué. Ces étapes se répètent jusqu'à ce que chaque neurone ait reçu l'erreur correspondante. Se basant sur ce signal d'erreur, le poids de chaque connexion est mise à jour afin de converger vers un état qui permet d'avoir une représentation interne de tous les modèles. Cette phase est nommée selon J.A. Freeman et D.M. Skapura (1991) : ***Propagation-Adaptation.***

Illustrons ce fonctionnement avec le précédent exemple des cours de bourse. Nous souhaitons obtenir de notre réseau de neurones la reconnaissance du modèle suivant :

Lundi=40, Mardi=20, Mercredi=40, Jeudi=60, Vendredi=80

Ce modèle correspond aux cours de la figure 25. Le réseau de neurones va premièrement devoir apprendre.

Nous lui fournissons en entrée les cours de la figure 27 :

Lundi=40, Mardi=20, Mercredi=**30**, Jeudi=**70**, Vendredi=80

Il transmet ces données en entrée à ces couches cachées qui procèdent à des modifications sur les cours. Le réseau n'ayant encore aucun modèle de référence, ce premier résultat peut être totalement aléatoire.

Voici une figure illustrant la transmission des cours de bourse à travers le réseau.

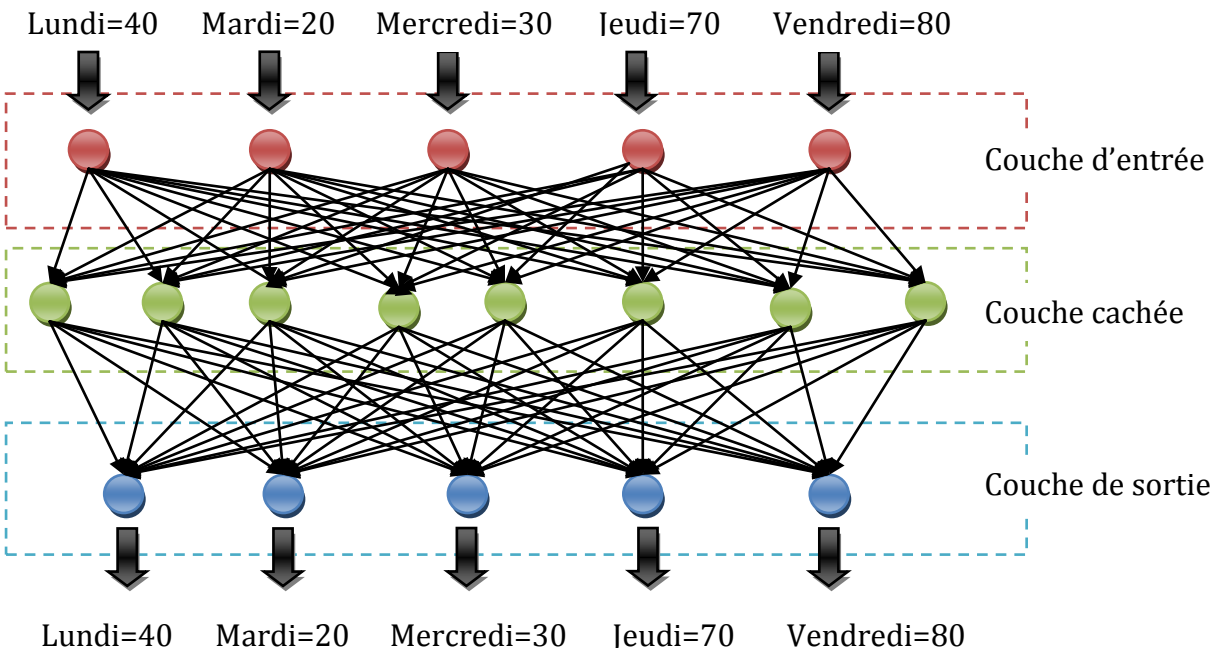


Figure 25 - Apprentissage d'un réseau de neurones

Ici nous supposons que le résultat de la couche de sortie est identique à la couche en entrée plutôt qu'aléatoire.

Nous fournissons maintenant au réseau de neurones le modèle de cours de bourse idéal (figure 26) afin qu'il le compare à son résultat et en ressorte des signaux d'erreurs. Ces erreurs qui sont les cours de Mercredi=30 au lieu de Mercredi=40 et Jeudi=70 au lieu de Jeudi=60 sont ensuite retransmises aux couches intermédiaires pour que les neurones corrigent les poids de leurs interconnexions. La figure suivante présente ce fonctionnement (les poids présentés ont des valeurs totalement arbitraires et une partie des neurones de la couche cachée a été enlevée pour améliorer la lisibilité du schéma).

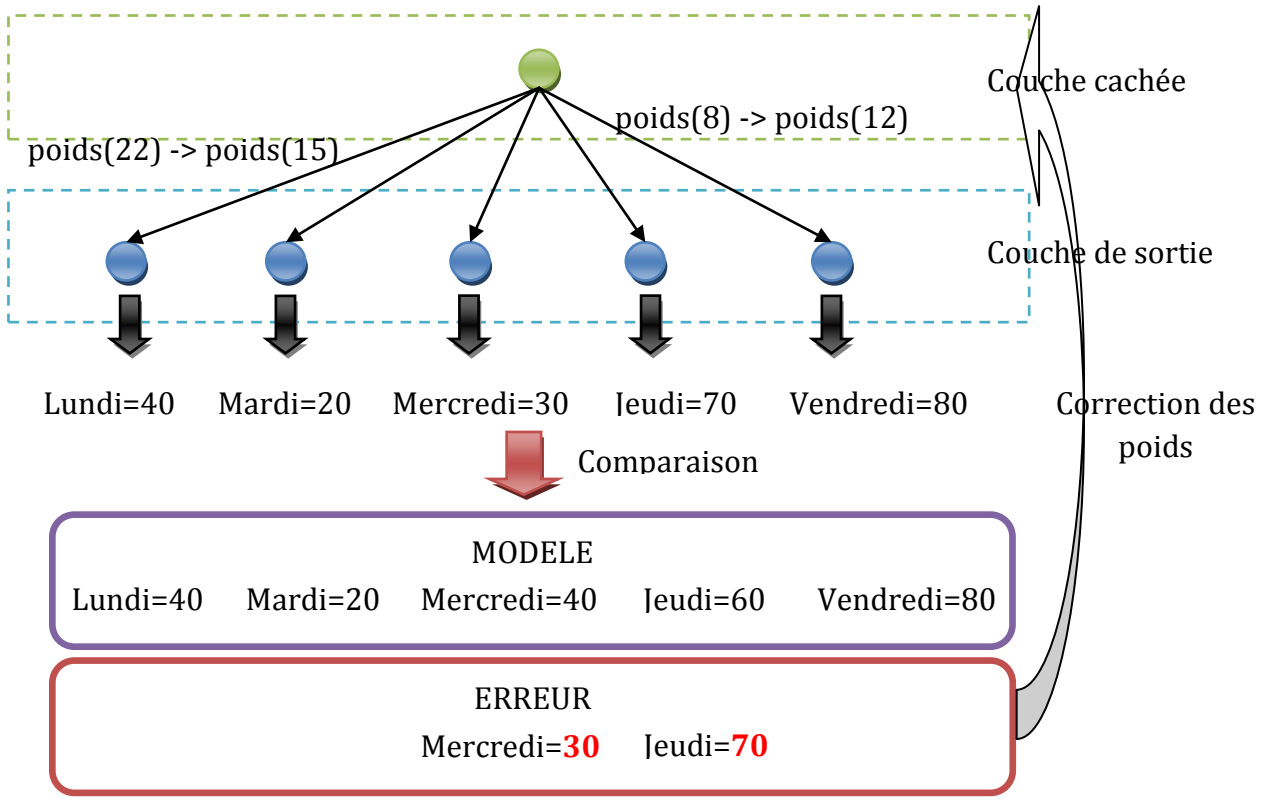


Figure 26 - Correction des erreurs d'un réseau de neurones

Ainsi les neurones se sont organisés de façon à reconnaître les cours de bourse pris en entrée. Après cet entraînement, si on présente au réseau des cours de bourse contenant du bruit, les neurones de la couche cachée seront en mesure de trouver le modèle correspondant. Cependant, si les cours en entrée sont trop éloignés des modèles qui ont servi à l'apprentissage, les résultats du réseau peuvent être totalement aléatoires.

Cet exemple nous démontre l'utilité de la rétro-propagation appliquée au réseau de neurones. Il palie au problème de traitement de données bruitées ou incomplètes.

Notre deuxième problème est la génération de cours de bourse à partir des cours actuels. Nous pouvons voir ce problème comme étant une suite de cours de bourse

incomplète. En reprenant l'exemple précédent, nous fournissons au réseau de neurones le modèle que nous souhaitons qu'il apprenne. Soit la suite :

Lundi=40, Mardi=20, Mercredi=40, Jeudi=60, Vendredi=80

À partir de ce modèle, nous prenons comme hypothèse que nous venons de recevoir le cours de mardi et que nous devons connaître la probabilité des cours des jours suivants.

Voici la figure suivante qui illustre le problème :

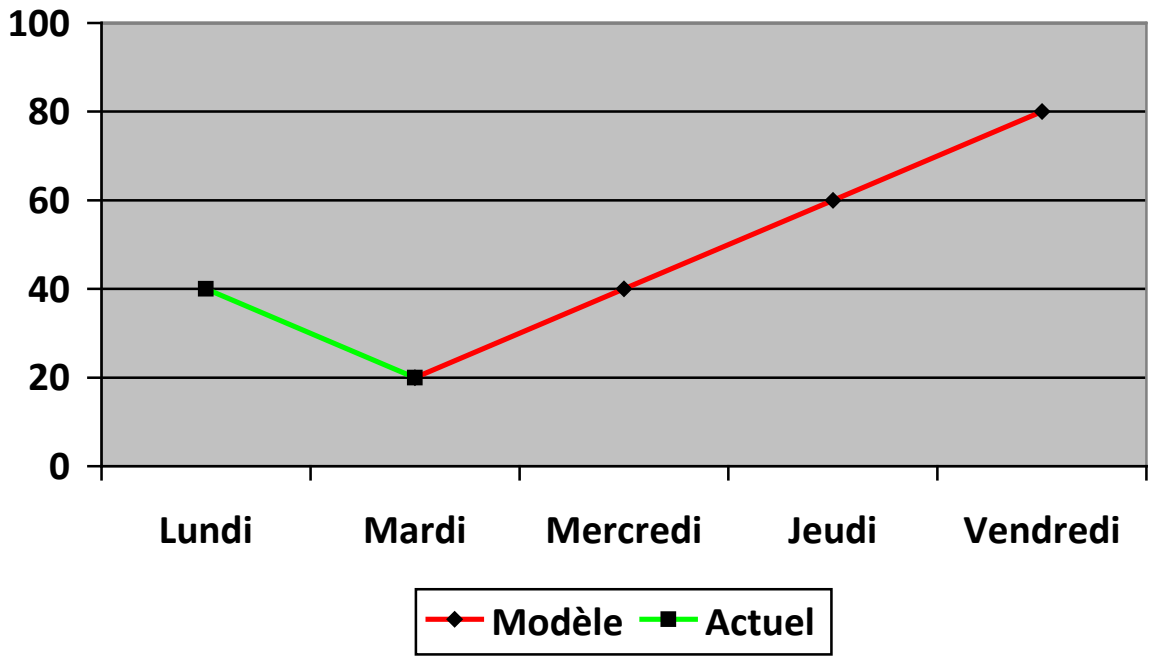


Figure 27 - Graphe de génération d'une suite de cours de bourse incomplète

En vert le cours actuel aujourd'hui, soit : Lundi=40, Mardi=20, et en rouge le cours attendu.

Lorsque le réseau de neurones prendra en entrée les deux cours de lundi et mardi, les neurones de la couche cachée feront le parallèle entre le modèle précédemment fourni et les cours incomplets. Le résultat en sortie sera le modèle que nous avions fourni.

Le réseau de neurones nous permet de résoudre nos deux problèmes : trouver dans l'historique des cours de bourse celui qui se rapprochent le plus de la situation actuelle, et fournir un modèle de cours après une suite incomplète.

Maintenant que nous avons expliqué la fonction de rétro propagation, nous allons devoir expliquer la fonction de chaque composant du réseau de neurones.

Les couches.

Parmi les typologies de réseau qui existent, nous avons expliqué pourquoi nous choisissons un réseau de neurones de type multicouches. Le réseau de neurones multicouches consiste en une, deux ou trois couches de neurones. Une couche de neurones a un nombre arbitraire de neurones. La donnée en sortie d'un neurone d'une couche est la donnée en entrée d'un neurone de la couche suivante. Cette donnée est le résultat de la somme des poids de toutes les synapses lui fournissant la donnée. Nous avons donné assez de schémas précédemment pour ne plus avoir à présenter schématiquement le réseau de neurones multicouches. La difficulté réside dans le fait de définir le nombre de couches et le nombre de neurones par couche.

Étant multicouches, le réseau de neurones contiendra au minimum une couche d'entrée pour recevoir les données, une couche cachée pour le calcul et une couche de sortie pour présenter les résultats. L'on pourrait en déduire que c'est un réseau comportant trois couches. Cependant B. Krose et P Smagt (1996) expliquent qu'étant donné que la couche d'entrée ne réalise aucun calcul, cette couche n'est pas comptée. Nous sommes donc sur un réseau à deux couches (minimum).

Le nombre de neurones dans chacune de ces couches va dépendre de différents paramètres.

La couche d'entrée est la couche qui va être la porte d'entrée des données au réseau de neurones. Dans notre cas, c'est grâce à cette couche que nous allons pouvoir transmettre notre portion de cours de bourse afin d'en avoir une prédiction. Chaque neurone de cette couche prendra en entrée un seul cours. Chaque cours d'une série sera transmise à un neurone. C'est-à-dire que si nous souhaitons que le réseau de neurones calcule une prédiction d'après trois cours de bourse, il faudra trois neurones à la couche d'entrée. Ces trois cours seront transmis par la couche d'entrée à la couche cachée via les synapses.

La couche de sortie est au contraire la porte de sortie du réseau de neurones, c'est la couche qui va nous présenter le résultat. Chaque neurone de cette couche va nous présenter un résultat correspondant à un cours. Cet ensemble de neurones, va présenter un ensemble de résultats résultant en une série de cours. Cette série de cours est la prédiction. Pour exemple, si nous souhaitons avoir une prédiction de 2 jours à raison d'un cours par jour, il nous faudra paramétrer la couche de sortie pour avoir 2 neurones et ainsi une série de 2 cours.

Les exemples de la couche d'entrée et de la couche de sortie sont représentés sur la figure suivante :

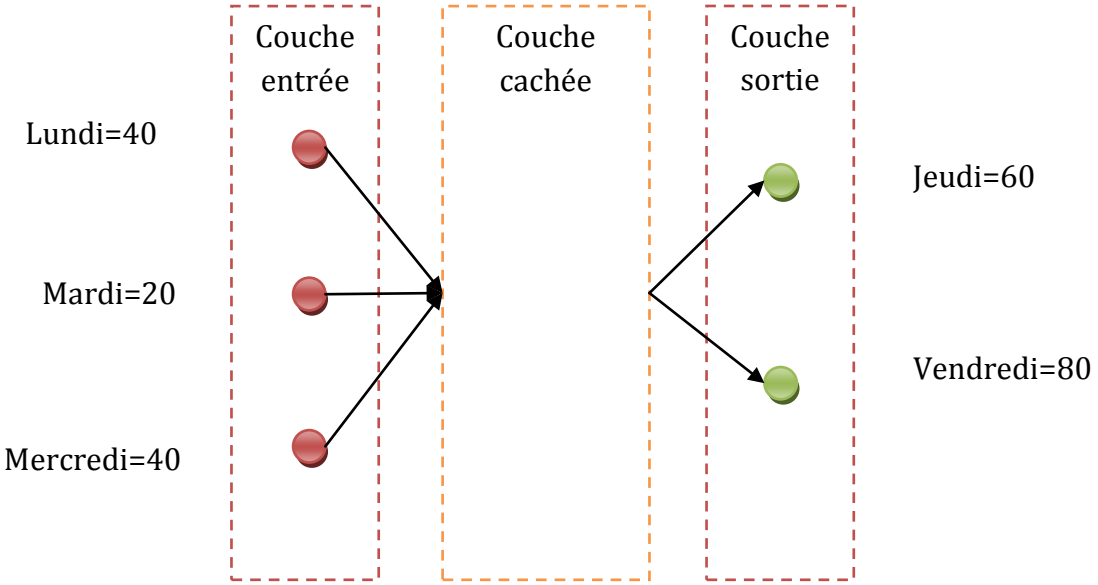


Figure 28 - Schéma du réseau de neurones multicouches avec 3 neurones en entrée et 2 en sortie

Le nombre de neurones contenus dans la couche d'entrée et de sortie se décidera en fonction de la stratégie d'investissement que nous adopterons. Ainsi pour une stratégie d'investissement de plusieurs jours, le nombre de cours en entrée devra être suffisant pour que le réseau de neurones puisse trouver le modèle optimal.

Nous précisons le paramétrage de ces couches dans le paragraphe sur l'apprentissage.

La couche cachée va être le cœur du réseau de neurones. C'est cette couche qui va réaliser la plupart des calculs. Elle sera composée au même titre que les autres couches d'un ensemble de neurones. Elle prendra en entrée les données provenant de la couche d'entrée et enverra ces résultats à la couche de sortie. Le nombre de neurones contenus dans cette couche n'est pas fixe et ne dépend pas de critères externes comme la stratégie d'investissement. Ce nombre de neurones est évalué par M. Zhang (2009) sans pour autant le justifier entre la moitié des neurones de la couche d'entrée et son double. Pour réussir à évaluer le nombre de neurones nécessaires à la couche cachée, nous allons reprendre l'exemple donné par B. Krose et P Smagt (1996). Dans leur exemple ils distinguent le modèle de cours, du cours idéal. Le modèle de cours est un ensemble de points stratégique du cours idéal.

La figure suivante illustre notre exemple. Dans les deux graphes A et B le cours idéal est représenté par des pointillés, le modèle par des cercles et la prédiction par une ligne

continue. Dans le réseau de neurones ayant généré la prédiction du graphe A (à gauche), la couche cachée contient 5 neurones. La couche cachée du réseau du graphe B (à droite) en a 20. Nous pouvons remarquer que sur le graphe B, la prédiction est en parfaite adéquation avec le modèle représenté par les cercles. Mais elle est très écartée de l'idéal. Cet effet s'appelle le *surentraînement*. Ce cas arrive lorsque les modèles d'entraînement contiennent beaucoup de bruit et que le réseau de neurones essaye de reproduire au maximum ce bruit plutôt que de donner une approximation.

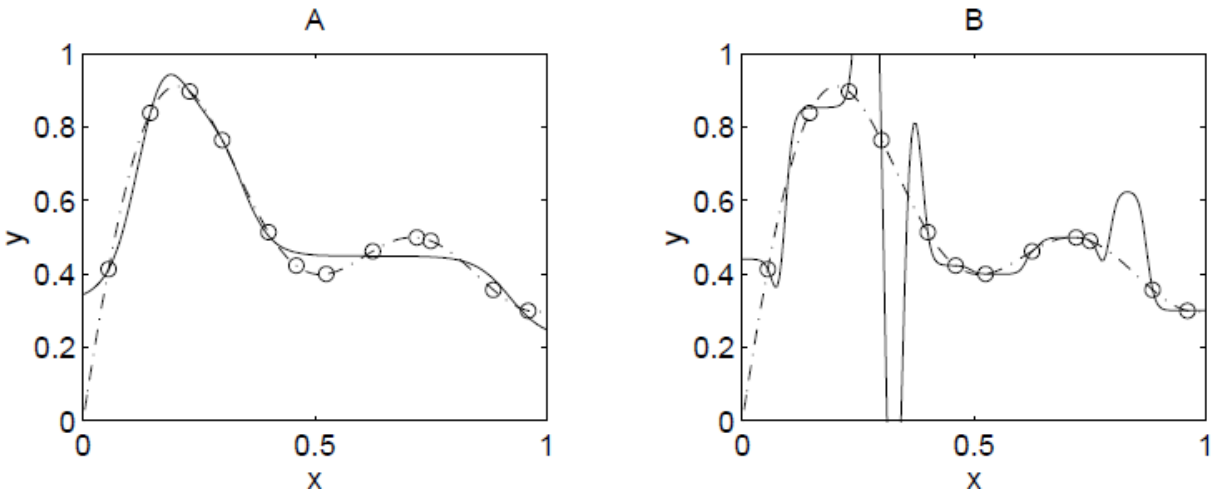


Figure 29 - Graphes représentant la différence entre deux prédictions avec des nombres de neurones dans la couche cachée différents

Les synapses

Cet exemple démontre qu'un nombre important de neurones de la couche cachée peut donner lieu au surentraînement. Lors de la phase d'apprentissage le taux d'erreur étant très faible (la prédiction étant en parfaite adéquation avec le modèle), cependant la prédiction est loin de la réalité.

Nous ajusterons donc le nombre de neurones de la couche cachée en nous basant sur ces faits. Malgré tout, cet ajustement ne pourra se faire que par des tests.

Le réseau de neurones multicouches peut aussi être composé de plusieurs couches cachées. Le principal désavantage est la capacité de calcul. Plus il y a de couches et de neurones, plus il aura de calculs à faire. Lippmann (1988) a démontré que pour un problème de classification, et avec un nombre de neurones *d*, la première couche cachée fonctionne souvent comme des hyperplans qui partitionnement efficacement un espace de dimension *d* en région. Et chaque neurone dans la couche cachée suivante va représenter une grappe de points (en anglais « cluster ») appartenant à la même classe.

Le tableau suivant a été réalisé à partir du document de Lippmann (1988) pour représenter les espaces de décisions en fonction du nombre de couches cachées :

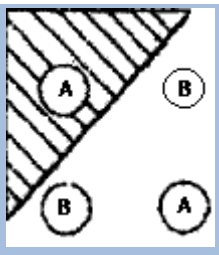
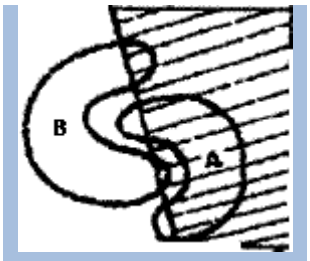
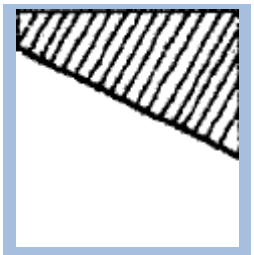
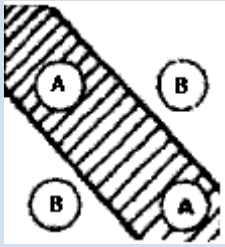
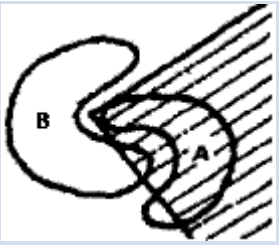
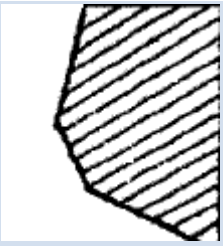
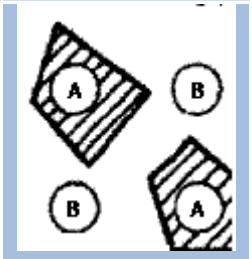
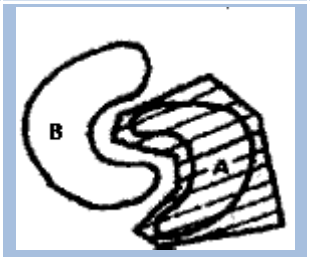
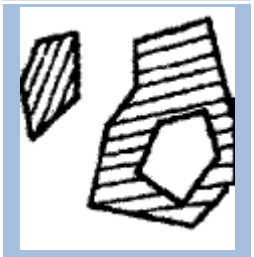
Structure	Type d'espace de décision	Exclusif ou problème	Classe avec région	Espace courant des régions
Une couche (Perceptron)	Moitié d'un plan délimité par l'hyperplan			
Deux couches	Région convexe, ouverte ou fermé			
Trois couches	Arbitraire (complexité limité par le nombre de neurones)			

Figure 30 - Tableau définissant les régions que peuvent représenter un réseau de neurones à rétro propagation

Les neurones et les synapses

Nous avons vu dans la partie fonctionnelle que la synapse servait à lier les neurones entre eux. Les synapses peuvent être avec poids, sans poids, un à un, ou direct. Notre réseau de neurones doit se souvenir de ces expériences passées, dans notre cas au moyen de l'apprentissage. Les synapses avec des poids nous permettent de créer ce « souvenir » dans le réseau. Pour comprendre comment ce souvenir est créé, faisons une analogie avec le monde animal. Les animaux apprennent avec le temps qui passe en se basant sur les informations de leur environnement. Chaque interaction avec

l'environnement peut être vue comme une mesure de la performance du système, et résulte d'un petit changement dans le système qui améliore les performances. Si le déplacement des membres dans une direction mène vers la nourriture, alors cela renforce le comportement de l'animal en fonction des données en entrée.

Le même principe est appliqué au réseau de neurones. Si augmenter le poids d'une synapse particulière mène à diminuer la performance ou augmenter l'erreur, alors le poids est diminué.

Le nombre de changements réalisé à chaque étape de l'entraînement sur les poids est très petit pour être sûr que le réseau ne s'éloigne pas trop de l'évolution de son état (ou de son état précédent). Mais si le nombre de changements est trop petit, alors le réseau nécessitera une longue période d'entraînement.

Quant aux neurones ils ont pour responsabilité d'exécuter la *fonction de sommation* et la *fonction d'activation*.

Cette fonction du neurone qui s'appelle la fonction de *sommation* va réaliser la somme des produits des données et des poids des synapses qui a servi à transmettre les données. Les données *d* en entrée lui sont transmises par la couche précédente et le poids lui est transmis par la synapse le rattachant à la couche précédente.

La figure suivante schématise le neurone et sa fonction de sommation :

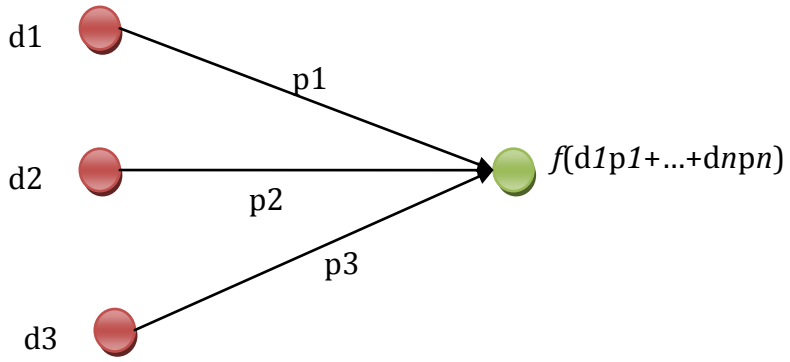


Figure 31 - Schéma de la fonction de sommation d'un neurone

Quelques réseaux de neurones ne réalisent pas la somme des produits mais le produit des produits tels que les réseaux « *Sigma-pi ».* Ce type de réseau est intéressant pour les typologies de type *High Order Neural network* qui, d'après M.Zangh (1996), devrait faire l'objet d'étude approfondie.

La deuxième fonction du neurone est la fonction d'*activation*. Elle a pour rôle de décider si elle doit transmettre le résultat de la fonction de sommation. Cette fonction correspond à l'influx nerveux d'un neurone biologique (ou potentiel d'action). Ce potentiel d'action d'un neurone biologique va transmettre la somme des « *potentiels gradués* » des synapses (fonction de sommation) si cette somme dépasse un seuil d'excitabilité du neurone. En dessous de ce seuil, cette somme n'est pas transmise. La fonction d'activation va agir de la même façon. Comme nous l'avons vu dans le chapitre des spécifications fonctionnelles, plusieurs fonctions d'activation existent. La plus courante est la fonction sigmoïde (figure 35).

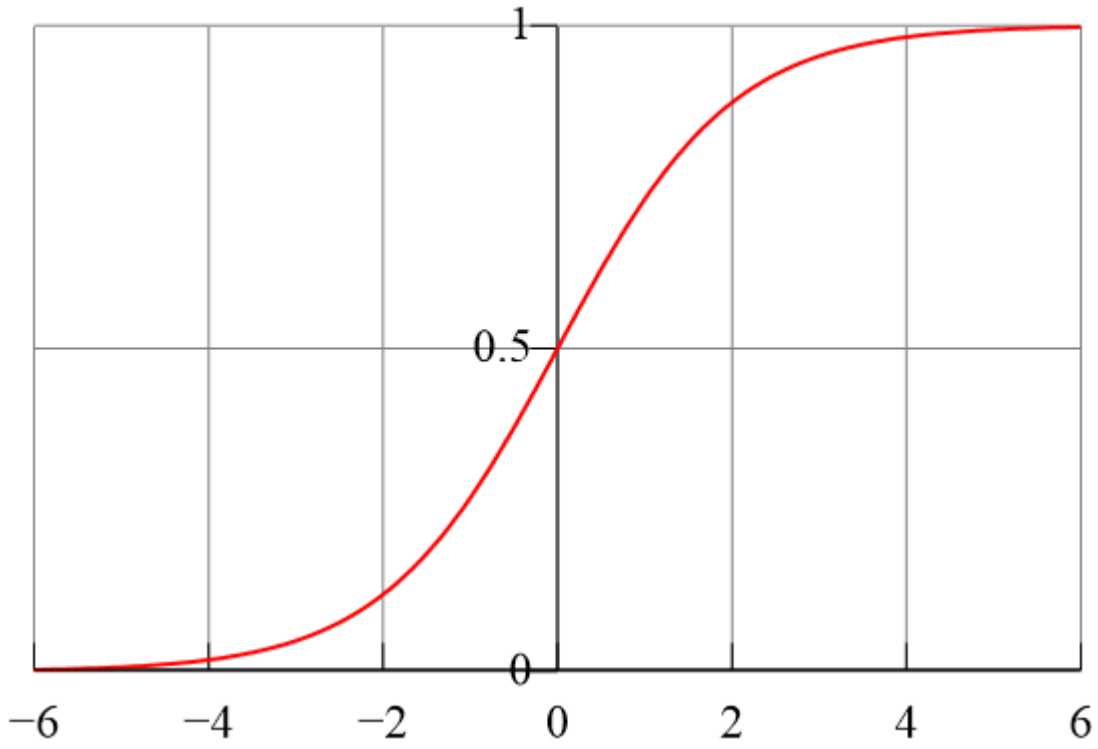


Figure 32 - Fonction d'activation sigmoïde

D'après K.Mehrotra, C. K. Mohan et S. Ranka (1996) l'avantage de cette fonction est que sa finesse rend particulièrement simple la fonction d'apprentissage. Voici la fonction sous forme de formule mathématique :

$f(net)=1/(1+exp(-2(net)))$

Où ***net*** représente le résultat de la fonction de sommation.

D'après L. Fausett (1993), la fonction d'activation doit être la même pour tous les neurones peu importe la couche. Nous partirons de ce principe là pour la configuration du réseau.

L'apprentissage

La propriété la plus importante dans un réseau de neurones est sa capacité à apprendre de son environnement et l'amélioration de ces performances avec l'apprentissage. Comme nous l'avons décrit avec le processus de rétro propagation, le réseau de neurones apprend de son environnement et ajuste les poids des synapses. Idéalement, le réseau devient au fur et à mesure plus *intelligent* après chaque itération du processus d'apprentissage. Mendel and McClaren (1970) définissent l'apprentissage d'un réseau de neurones comme :

« A process by which the free parameters of a neural network are adapted through a process of stimulation by the environment in which the network is embedded. The type of learning is determined by the manner in which the parameter changes take place. »

Que l'on peut traduire par :

« Un processus par lequel les paramètres non-statiques d'un réseau de neurones sont adaptés à travers un processus de stimulation par l'environnement dans lequel le réseau est embarqué. Le type d'apprentissage est déterminé de la façon dont les changements des paramètres prennent place ».

Cette définition de l'apprentissage implique les processus suivants :

1. Le réseau de neurones est stimulé par son environnement.
2. Le réseau de neurones subit des changements de ses paramètres non-statiques dû à la stimulation.
3. Le réseau de neurones répond d'une nouvelle façon à l'environnement suite aux changements qui ont eu lieu dans sa structure interne.

L'ensemble de règles qui définissent l'apprentissage est appelé un *algorithme d'apprentissage*. Cet algorithme d'apprentissage est, dans notre cas l'algorithme de rétro-propagation. Nous avons expliqué précédemment comment celui-ci fonctionnait.

Dans notre cas, l'environnement du réseau de neurones est les cours de bourse qui arrivent au fil de l'eau. Pour réussir à stimuler le réseau de neurones, il doit être paramétré afin que les cours de bourse en entrée produisent les données désirées en sortie. Les synapses, et plus particulièrement le poids associé aux synapses, remplissent

cette fonction de paramétrage. Plusieurs méthodes existent pour ajuster les synapses, ou (pour reprendre la définition utilisée dans les processus), stimuler les synapses. L'une d'entre elles consiste en l'ajustement des synapses avec une matrice de poids préétablie. Une autre est d'entraîner le réseau en l'alimentant avec des modèles et en le laissant ajuster le poids des synapses. Ces modèles d'apprentissage sont nommés par S.Haykin (2005) : *paradigme d'apprentissage*. Nous allons utiliser cette dernière méthode car comme nous allons le démontrer, nous avons besoin d'établir des *paradigmes* représentatifs de certains comportements boursiers sur lesquels le réseau de neurones va pouvoir s'appuyer.

Le but du système que nous élaborons, va être de réaliser un maximum de profit sur une très courte période. C'est sur cette période de temps que nous allons devoir établir nos paradigmes afin d'entraîner le réseau de neurones. W. Remus et M. O'connor (2001) suggère quelques principes pour définir ces paradigmes :

- Nettoyer les données.
- Mettre les données à l'échelle et les désaisonnaliser.
- Utiliser une méthode appropriée pour choisir un bon point de départ.
- Utiliser une méthode spécialisée pour éviter l'optimum local.

Nous les avons donc adaptés à nos besoins en réalisant ces étapes :

1. Définir une période de temps.
2. Découper l'historique des cours de bourse en échantillon d'une durée égale à la période précédemment définie (« choisir un bon point de départ »).
3. Enlever le bruit de ces cours (« nettoyer les données »).
4. Entraîner le réseau avec ces paradigmes (« éviter l'optimum local »).

Nous avons besoin de définir une période de temps, afin de trouver les modèles sur cette période-là. Cette période de temps peut être définie de deux façons : par stratégie (on ne souhaite qu'investir sur du très court terme pour diminuer le risque) ou par la réalisation de tests à l'aide du réseau de neurones (par exemple une période de 10 minutes peut de contenir plus de comportements similaires qu'une période de 2 secondes). Nous choisirons dans un premier temps de choisir cette période de façon stratégique, car le nombre de tests à réaliser est relativement important).

Pour le deuxième point nous découpons l'historique des cours de bourse par période pour en réaliser des modèles. Le graphique suivant illustre le découpage de cours sur une période de vingt minutes:

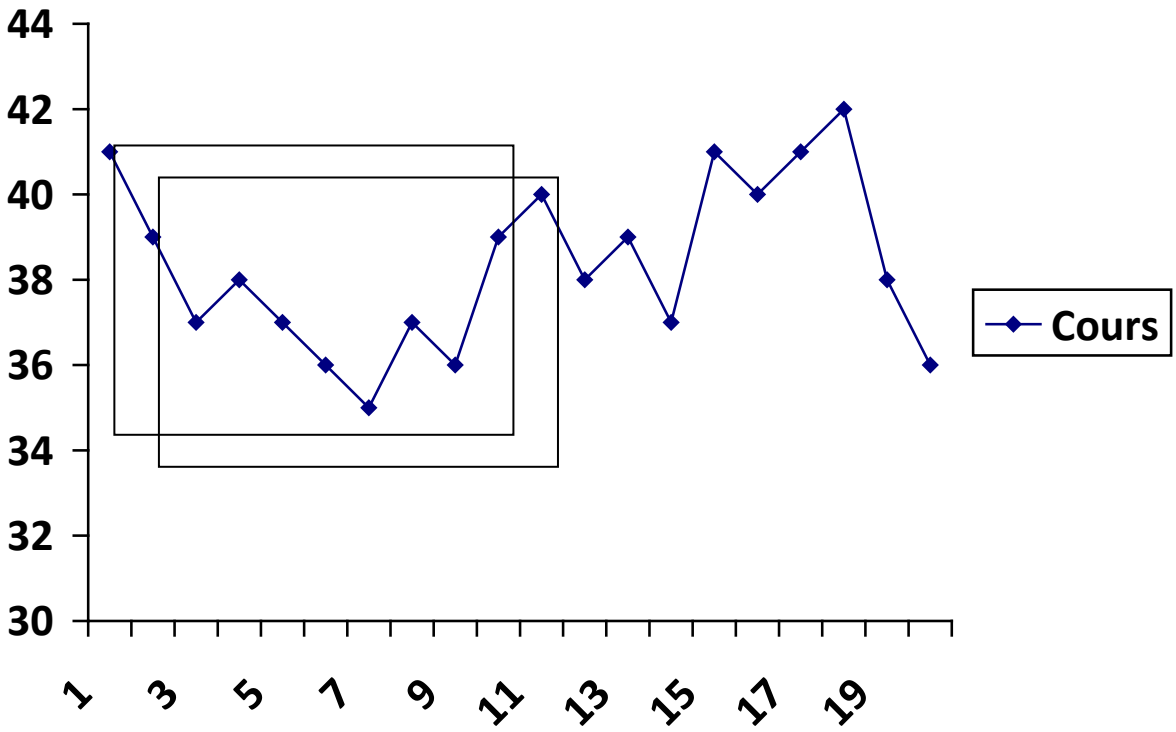


Figure 33 - Extraction des modèles d'un cours de bourse

Les deux carrés qui entourent les cours représentent chacun un modèle de cours de 10 minutes espacés d'une minute. Ce sont ces deux modèles (et les modèles suivant décalés d'une minute à chaque fois. non représentés sur le graphique) qui seront fournis au réseau de neurones pour son apprentissage.

La troisième étape est le nettoyage des données. Comme nous l'avons vu, nous allons devoir enlever le bruit des modèles que nous avons extrait de l'historique des cours de bourse. Pour cela, nous allons utiliser une simple technique d'extraction des points qui nous semblent intéressants pour atteindre nos objectifs. Ces points vont être les extrémités d'une hausse ou d'une baisse si elles ont été suffisamment grandes pour qu'elles nous soient profitables. En reprenant le deuxième modèle du graphique précédent nous notons chaque point avec une lettre :

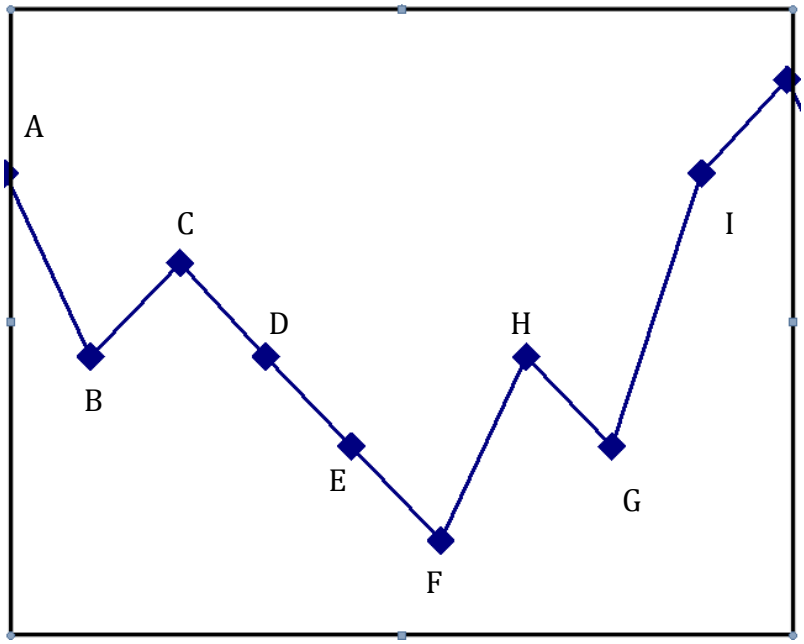


Figure 34 - Graphique pour l'extraction des points

Ce graphique issu d'un historique de cours va nous permettre d'en extraire un modèle. Si nous prenons comme exemple que les frais dus à un ordre en bourse sont de 1 point (C'est-à-dire que si nous achetons il faut que le cours monte au minimum de 2 points pour qu'il nous soit profitable). Ainsi sur notre graphique, nous décidons de vendre au point A. Le cours descend jusqu'à B. Nous pouvons à ce moment-là envisager de racheter nos actions vendues au point A pour réaliser le bénéfice B-A. Nous pouvons aussi envisager de racheter des actions pour faire un profit jusqu'au point C. Cependant, comme nous l'avons mentionné, les frais d'ordre sont de 1 point. C'est-à-dire qu'acheter en B n'engendre aucun profit. Ni même de racheter en B et vendre en C. Ainsi il est plus intéressant de garder la position que nous avons adoptée en A. Ensuite le cours descend jusqu'à F. À ce moment-là, il devient intéressant de racheter les actions vendues en A. Ainsi le bénéfice sera :

(F-A) - 1 point de frais

Soit 2 points (en supposant que A et C soient de valeurs égales)

Nous appliquons la même méthode ensuite de F jusqu'à I et plus. L'intervalle H/G n'est pas intéressant car de trop faible intensité. Nous aurons donc (I-F)-1point.

Au final notre modèle sera celui-ci :

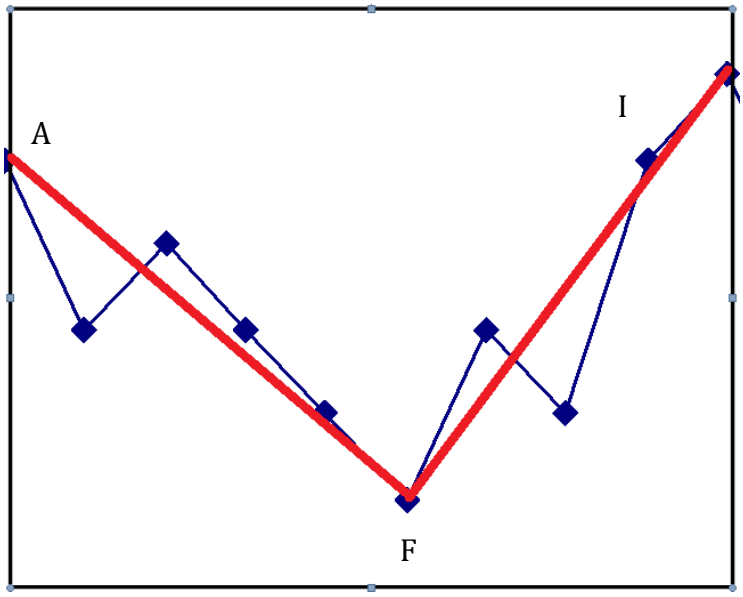


Figure 35 - Graphique représentant le modèle après extraction des points

La ligne rouge représentant le modèle extrait des cours de bourse.

Cette méthode a l'avantage de prendre en compte les frais de bourse. Bien que cela puisse être aussi le pourcentage escompté pour chaque position prise (ex : nous souhaitons un bénéfice de 3 points par position). Ainsi en ajustant nos modèles, nous pourrons entraîner différents réseaux de neurones avec différents taux de rentabilité à la clé.

La quatrième et dernière étape est l'apprentissage du réseau de neurones. Il va falloir lui fournir les données appropriées que nous avons travaillées précédemment pour qu'il s'auto-corrige. C'est-à-dire comme nous l'avons vu maintes fois, il ajustera le poids de ces synapses en fonction des erreurs qu'il fera.

Les paramètres que nous devrons lui fournir sont :

- Le cours de bourse sur une période de temps (ex : figure 37)
- Le modèle de cours que nous avons filtré (ex : figure 38)

Ainsi le réseau après une phase d'initialisation, prendra le cours de bourse, le fera traverser ses différentes couches et neurones, puis en sortira un résultat. À partir de là entrera en compte l'algorithme de rétro propagation. Cet algorithme comparera le résultat avec le modèle de cours fourni, et en calculera une erreur où un écart. De cet écart, il s'en servira pour ajuster ses poids au fur et à mesure de l'entraînement.

L'évaluation des performances

L'évaluation des performances permet d'évaluer et de valider une prédiction du réseau de neurones. Cette évaluation va nous permettre de connaître le niveau de fiabilité des prédictions du réseau de neurones. Elle passera par le calcul d'un écart entre le résultat et le cours réel. La validation se fera par l'appréciation de cet écart.

Notre attention se portera sur des méthodes d'évaluation dédiées à la prédiction de cours de bourse. Ces méthodes permettent non seulement d'évaluer l'écart, mais de calculer le sens directionnel de l'écart. Ainsi un écart entre un cours de bourse qui descend et une prédiction de cours de bourse qui monte sera plus important que si les deux cours, le cours réel et le cours prédit, vont dans la même direction. D'après M.Zhang (2009), les méthodes généralement utilisées pour l'évaluation des performances sont :

- La valeur efficace ou moyenne quadratique.
- Coefficient de corrélation.
- Pourcentage du maximum absolu.
- Pourcentage de la moyenne absolue.

Ces méthodes donnent une bonne évaluation quant à l'écart entre deux séries de valeurs. Dans notre cas, nous avons besoin d'inclure un sens directionnel. Pour illustrer notre problème, les graphiques suivants comportent chacun un cours réel et une prédiction :

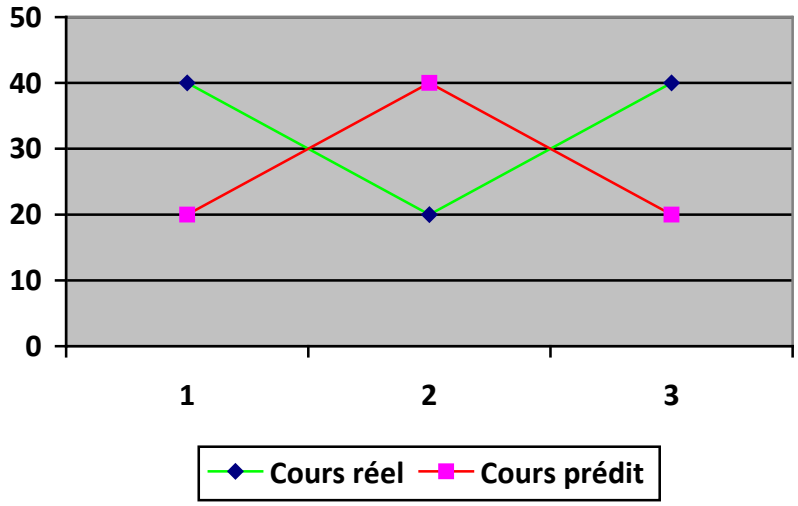


Figure 36 - Graphique représentant une prédiction incorrecte

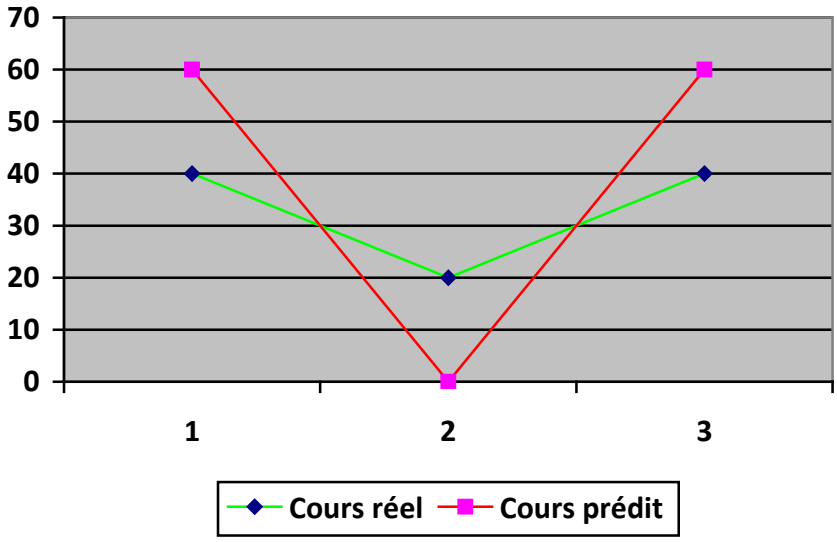


Figure 37 - Graphique représentant une prédiction incorrecte mais avec un sens directionnel correct

Dans le premier graphique, le cours prédit est éloigné du cours réel et il ne va pas dans la même direction. Dans le deuxième graphique le cours prédit est aussi éloigné, mais il va dans la même direction. De notre point de vue, le deuxième graphique représente une situation bien plus intéressante que le premier. Notre position sur le deuxième graphique aurait été de vendre en 1, racheter en 2, puis vendre en 3. Il y aurait juste eu une différence de bénéfice.

Si nous appliquons à ces deux graphes les méthodes d'évaluation des performances précédentes, nous obtenons les valeurs suivantes:

	Cours réel	Graphique 1	Graphique 2
La valeur efficace ou moyenne quadratique	34.64	28.28 (diff. 6.35)	48.98 (diff. -14.34)
Coefficient de corrélation		-1	1
Pourcentage du maximum absolu		100(%)	100(%)
Pourcentage de la moyenne absolue		66.66(%)	66.66(%)

A la lecture du tableau, nous pouvons constater que seules les méthodes de valeur efficace et de coefficient de corrélation démontrent une différence entre les deux cours. Toujours d'après M.Zingh (1999), la méthode généralement employée pour évaluer correctement une prédiction est la méthode de la Moyenne Normalisée de la Racine carrée de l'Erreur (en anglais « Normalized Mean-Square Error » ou NMSE). Toutefois dans son exemple sur la prédiction des taux d'intérêts étrangers, exemple très similaire à notre étude, il suggère d'utiliser dans les méthodes :

- La moyenne absolue de l'erreur.
- La symétrie directionnelle.

En appliquant ces deux méthodes à notre exemple précédent pour les comparer aux méthodes précédentes, nous obtenons ceci :

	Graphique 1	Graphique 2
La moyenne absolue de l'erreur	20	20
La symétrie directionnelle	0 (%)	100 (%)

La moyenne absolue de l'erreur nous montre que l'écart entre le cours de prédiction et le cours réel des deux graphiques est identique.

La symétrie directionnelle nous montre que le cours de prédiction du graphique 1 est totalement inversé par rapport au cours réel. Et elle nous montre que le cours de prédiction du graphique 2 progresse exactement dans la même direction que le cours réel.

Nous utiliserons donc la combinaison de ces deux méthodes pour évaluer la qualité des prédictions du réseau de neurones.

Nous avons détaillé dans cette partie le module de réseau de neurones d'un point de vue technique. Nous remarquons que c'est un module plus complexe que les autres. Nous devrons, pour améliorer sa qualité, réaliser de nombreux tests afin de trouver les paramètres adéquats.

3.2.5. LE MODULE DE GESTION DU RISQUE

Le module de gestion du risque va nous permettre de valider un signal (Achat/Vente), de valoriser le risque si nous prenons position, et, d'estimer à quelle hauteur nous pouvons nous exposer.

Pour pouvoir évaluer le risque, il faudra que le module de gestion du risque ait le signal généré par le module de génération du signal et le résultat d'analyse du réseau de neurones. Le premier permettant de savoir dans quel sens l'on doit prendre position, et le deuxième, d'estimer le risque. Ces deux résultats seront normalisés sous forme de message capable de transiter sur le bus de l'interface (nous avons décrit dans chaque module respectif, que chaque résultat d'analyse est posté sur le bus de l'interface). Une fois ces messages sur le bus, ils sont à disposition de tous les autres modules. Le module de gestion du risque sera intéressé par ces deux messages. Il s'abonnera donc au préalable au bus de l'interface en lui indiquant le type de message dont il souhaite être informé. Ainsi l'interface informera automatiquement lorsque les résultats (signal et analyse de l'IA) seront sur le bus.

Un premier problème se pose à cette manière de procéder : les messages sont asynchrones. En effet le résultat du module de génération du signal n'est pas forcément posté en même temps que le résultat du réseau de neurones. Bien que ces messages

doivent être postés sur le bus avec un intervalle réduit pour améliorer la réactivité du système, le module de gestion du risque doit être en mesure d'associer ces deux messages par le biais d'une date contenant l'heure, la minute, la seconde, millisecondes…. Pour cela, le module doit pouvoir gérer un cache très court terme. Il placera dans ce cache le premier message (ex : le signal) jusqu'à ce que le second message arrive (ex : le résultat de l'IA).

Le schéma suivant présente ce flux :

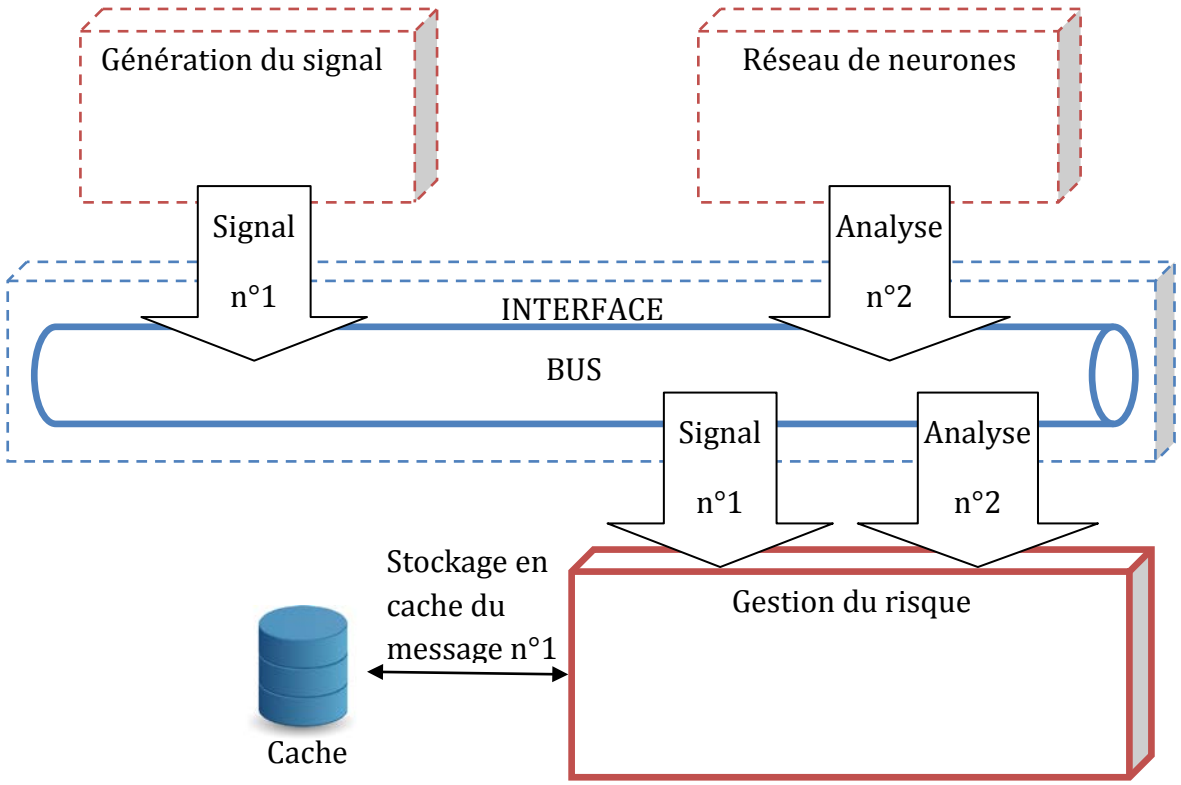


Figure 38 - Schéma de flux d'un signal allant au module de gestion du risque

Une fois la réception de ces messages gérée, le module peut débuter l'analyse. Comme décrit dans la partie fonctionnelle, le module utilisera la formule de VaR (Value At Risk) avec la méthode de variance-covariance pour mesurer le risque.

Cette formule va s'appuyer sur deux valeurs essentielles : le profit attendu et la durée d'exposition.

Le profit attendu et la durée d'exposition vont être calculés à l'aide du résultat du réseau de neurones et des statistiques fournis par le module de génération du signal.

Quand le réseau de neurones fera une prédiction, nous pourrons déduire un ratio de temps/rentabilité de cette prédiction.

Voici un exemple de prédiction du module du réseau de neurones sous forme de graphique :

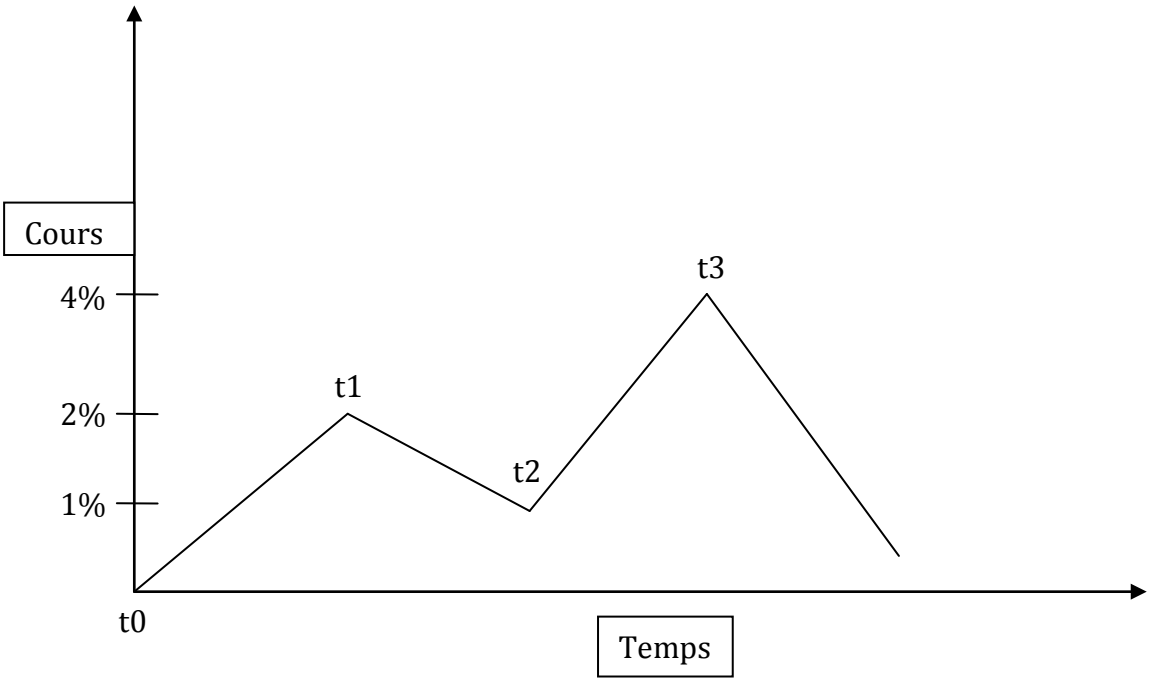


Figure 39 - Exemple de prédiction du réseau de neurones

Dans l'hypothèse où le module de génération du signal nous fournit un signal d'achat et le module du réseau de neurones nous fournit la prédiction ci-dessus, le module de gestion du risque va évaluer dans un premier temps la durée d'exposition. Dans notre schéma, si nous sommes acheteurs à t0, nous avons le choix de vendre à t1 ou à t3. Le temps t2 représentant un risque. Sachant que plus longtemps nous serons exposés plus le risque augmentera car plus la prédiction perdra en fiabilité.

Donc nous avons les choix suivants :

- Réaliser 2% sur une période t1
- Réaliser 4% sur une période t3

Nous avons ici nos deux valeurs nécessaires aux calculs de notre formule VaR.

Ainsi, nous pouvons évaluer le risque encouru grâce à la prédiction. Voyons maintenant comment évaluer ce risque grâce aux statistiques et au signal fourni par le module de génération du signal.

Le module de génération du signal va nous transmettre via son message un signal d'achat/vente, et les statistiques associées à ce signal. Nous aurons parmi ces statistiques la durée moyenne des positions de l'ensemble des stratégies qui ont permis de générer ce signal ainsi que leurs profits moyens. Nous retrouverons nos deux valeurs nécessaires au calcul de notre VaR.

A ce stade-là, nous nous retrouvons avec deux variables VaR qui, additionnées, vont nous permettre de connaître notre exposition globale au risque. Il nous reste à définir comment associer un montant à investir en fonction du risque.

Nous devrons définir une stratégie d'investissement. Il nous faudra établir combien nous souhaitons investir, quelle part de notre capital nous sommes prêts à perdre et combien de positions parallèles voulons nous avoir. Toutefois cette stratégie sera toujours en conséquence des risques définis précédemment.

Le risque défini, l'investissement calculé, le module de gestion du risque doit transmettre l'ordre à l'interface pour exécution. Pour transmettre cet ordre, il créera un message contenant les données utiles et générées par les différents modules au cours des processus précédents. Puis il placera ce message sur le bus de l'interface. Enfin l'interface pourra en prendre connaissance pour le transmettre au courtier.

Le schéma suivant illustre ce que nous venons d'expliquer :

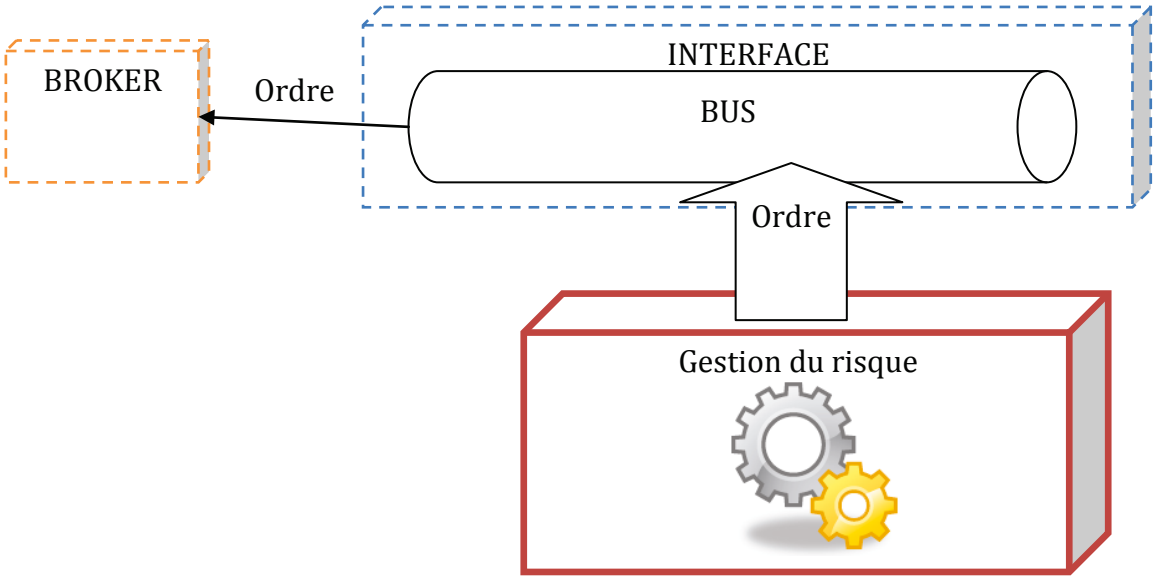


Figure 40 - Schéma de transmission d'un ordre

3.2.6. LE MODULE DE MONITORING/ADMINISTRATION

Le module de monitoring et d'administration aura pour rôle de présenter des informations à l'utilisateur et de lui permettre de configurer l'application et le système. Nous découperons ce module en deux parties : la partie métier et la partie système.

La partie métier aura pour but de gérer l'application d'un point de vue fonctionnel. Elle sera composée d'une sous-partie décisionnelle (monitoring) avec la présentation de données ayant un sens métier (exposition au risque, positions ouvertes…) et une partie configuration (administration) pour gérer les paramètres métiers (seuil de risque, frais des brokers…).

La sous-partie décisionnelle permettra de prendre des décisions cruciales pour l'amélioration des stratégies. Il nous faudra donc mettre en place un module complexe et performant de restitutions de données. Dans un but de délégation de la complexité applicative, nous opterons pour une solution décisionnelle externe. Parmi les solutions

existantes (Oracle PeopleSoft, IBM Cognos…) le logiciel JasperSoft (www.jaspersoft.com) nous paraît être une solution tout à fait adéquate car il nous permet de créer des rapports dynamiques et, en plus d'être open source et gratuit, il est facilement intégrable à une autre application.

La sous-partie configuration permettra de configurer l'application d'un point de vue métier. Elle devra être en mesure de modifier dynamiquement les paramètres de chaque module afin qu'ils soient pris en compte instantanément. Pour cela, chaque module devra être en mesure de fournir des fonctions d'accès au module d'administration.

La partie système présentera toutes les données liées au système d'un point de vue technique (état du réseau, consommation mémoire…), et permettra de paramétrer le système (base de données, réseau, serveur…).

3.2.7. TRANSVERSE

Parmi toutes les fonctions détaillées, nous ajoutons certaines fonctions qui assureront des tâches transverses au système. Parmi celles-là on trouvera :

- La fonction de sauvegarde

  Elle s'abonnera au flux de message transmis par l'interface pour les sauvegarder. Elle sauvegardera aussi tous les événements provenant des autres fonctions. Outre cela, elle devra mettre à disposition toutes ces données aux fonctions qui en auront besoin. Elle devra donc aussi assurer une fonction de service.

3.3. VALIDATION ET PERFORMANCE

Nous présentons dans cette partie les premiers résultats du prototype.

Les tests ont été réalisés en utilisant les cours réels de bourse. Les cours de bourse étant propriétaires (selon le diffuseur et les intermédiaires, les cours peuvent varier de façon très légère), il est difficile de se procurer un historique des données.

Nous nous sommes toutefois procurés les données sur les devises, notamment l'Euro contre le Dollar pour les périodes de :

- 08 novembre 1999 à aujourd'hui en valeur journalière. (3339 jours, ~400Ko) (l'euro étant côté à partir de l'année 2000, les valeurs précédentes sont par consensus prises auprès du deutschemark)
- 27 mars 2007 à aujourd'hui en *tick* (toutes les quelques secondes, jusqu'à plusieurs fois par seconde) (~3Go)

Un travail de lissage des données a été réalisé afin d'assurer une continuité au cours. Nous avons donc recréé à partir des données *tick* un cours seconde par seconde depuis le 27 mars 2007 à aujourd'hui.

Pour chaque donnée, nous avons :

- Le prix d'achat à l'ouverture.
- Le prix d'achat et de vente le plus haut.
- Le prix d'achat et de vente le plus bas.
- Le prix d'achat et de vente de fermeture.

Comme décrit plus bas, les tests sont réalisés sur un seul prix (à la seconde, à la minute…). Il est usuel de présenter un cours de bourse par son cours de fermeture. Donc pour n'avoir qu'un seul prix, nous ferons la moyenne entre le prix d'achat de fermeture et le prix de vente de fermeture :

Prix pris en compte = (prix d'achat de fermeture + prix de vente de fermeture)/2

Pour chaque test, nous présenterons la période utilisée.

3.3.1. AIDE À L'ANALYSE

L'aide à l'analyse nous permet d'exécuter nos premières stratégies et de tester le fonctionnement de ce module.

Nous allons valider le contexte nécessaire à l'analyste/développeur. Pour cela, nous aurons à valider :

- La connexion entre le système externe (Eclipse, Matlab ou R) et le module d'aide à l'analyse.
- La librairie de fonction disponible au développement de la stratégie.
- Le temps d'exécution.
- L'accès aux résultats d'exécution.

La première étape, la connexion entre le système externe et le module d'aide à l'analyse, a été testée avec Eclipse. Eclipse nous permet de rapatrier le contexte pour développer une stratégies sous forme de projet. Il nous permet ainsi d'accéder aux différentes librairies et fonctions de l'environnement de projet.

Cette première étape, n'ayant aucun point sensible ou difficulté particulière, se déroule sans aucun problème. Le rapatriement des librairies nécessaires au développement de stratégie se fait en quelques secondes. Leur utilisation est directement opérationnelle.

La deuxième étape consiste en l'utilisation de la librairie de fonction. Cette librairie contenant des centaines de fonctions (traitement de dates, chargement de données boursières, création des paramètres, fonctions mathématiques…) permet de développer la stratégie. Nous développons une première stratégie basée sur l'indicateur technique MACD (en anglais « Moving Average Convergence/Divergence » soit la convergence divergence des moyennes mobiles).

Cette stratégie se développe facilement en deux classes java. L'une comportant 194 lignes et l'autre 68, ce qui est relativement court. Cela démontre un panel de fonctions assez complet pour la librairie. Toutefois, les besoins peuvent grandement varier d'une stratégie à une autre, et, le manque de flexibilité quant à la possibilité d'enregistrement des résultats sera à améliorer.

MACD_Strategy.java | InterfaceLauncher.java | MarketDataHelper.java | Strategy.java | TimeFrameHelper.java | M

```java
package com.algotrade.anhel.strategy.ta.macd;

import java.util.ArrayList;

@Component
public class MACD_Strategy extends Strategy {

    private Logger log = LoggerFactory.getLogger(MACD_Strategy.class);

    public static void main(String[] args) throws Exception {
        start(MACD_Strategy.class);
    }

    @Override
    public SignalType execute(double[] quotes, AbstractParameters abstractParameters) throws Exception {

        MACDParameters macdParameters = (MACDParameters) abstractParameters;
        int optInFastPeriod = macdParameters.getFastPeriod();   //12; // Nombre de jours (< à SlowPeriod)
        int optInSlowPeriod = macdParameters.getSlowPeriod();   //26; // Nombre de jours (> à FastPeriod)
        int optInSignalPeriod = macdParameters.getSignalPeriod();//9; // Nombre de jour du signal

        if(quotes.length == 0)
            return null;

        int startIdx = 0; // Index de démarrage dans le tableau
        int endIdx = quotes.length-1; // Index de fin dans le tableau

        // Donnée sortie
        MInteger outBegIdx =        new MInteger();
        MInteger outNBElement =     new MInteger();
        double[] outMACD =          new double[endIdx-startIdx]; // Ligne MACD
        double[] outMACDSignal =    new double[endIdx-startIdx]; // Ligne de signal
        double[] outMACDHist =      new double[endIdx-startIdx]; // ?? Histogramme ?

        com.algotrade.anhel.indicator.MACD macd = new com.algotrade.anhel.indicator.MACD();
        try {
            macd.macd(startIdx, endIdx,
                    quotes,
                    optInFastPeriod, optInSlowPeriod, optInSignalPeriod,
                    outBegIdx, outNBElement,
                    outMACD, outMACDSignal, outMACDHist);
```

Figure 41 - Copie d'écran de la classe java représentant le code de la stratégie MACD

La troisième étape consiste en l'exécution de cette stratégie. Pour cela, la librairie de fonctions doit s'occuper de charger les données boursières en fonction des besoins de la stratégie, d'exécuter la stratégie, et d'enregistrer les résultats.

Après un premier lancement, le module d'aide à l'analyse évalue le temps que va prendre la stratégie pour s'exécuter avec tous les paramètres demandés. Il affiche le nombre total de stratégies à exécuter en fonction des paramètres, il donne le temps total d'exécution et demande confirmation. La copie d'écran suivante montre une stratégie en cours d'exécution :

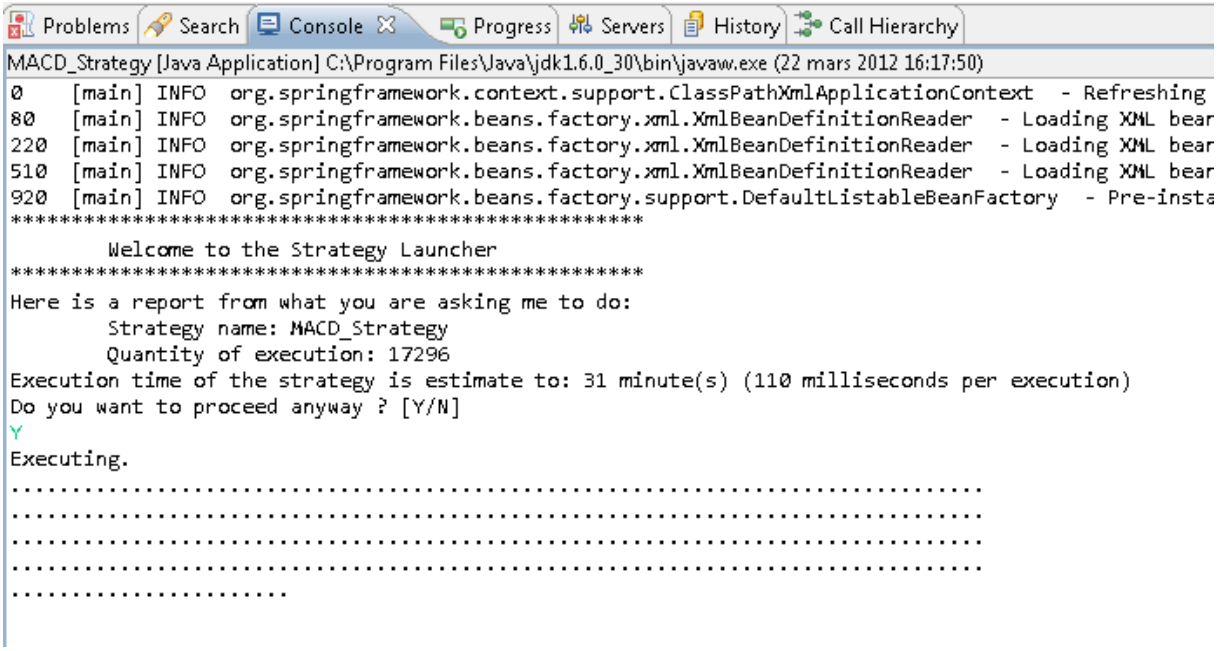


Figure 42 - Console Eclipse affichant l'état d'avancement d'exécution d'une stratégie

L'exécution nous affiche le nom de la stratégie « *MACD_Strategy* » la quantité d'exécution de stratégie « *17296* » (l'indicateur technique prend trois signaux en paramètre qui peuvent être différents suivant chaque exécution) et le temps d'exécution total « *31 minutes* ».

Le premier gros problème que nous rencontrons lors de l'exécution de la stratégie est le chargement des données. La stratégie en question s'exécute sur le cours de l'euro/dollar de 2003 à 2012 espacés d'une minute soit environ 1 200 000 données. Le chargement se fait en une seule fois donc le système est immédiatement impacté par l'occupation en mémoire de ces données.

Le deuxième problème est le temps d'exécution. Il y a certes un nombre important de stratégies à exécuter mais le fait d'attendre plusieurs minutes à chaque fois que l'on modifie la stratégie est trop long. Cela fait cher l'erreur de programmation.

La quatrième, et dernière, étape de contrôle est la qualité de présentation des résultats. Il faut que ces résultats donnent un bon état de la qualité de la stratégie mais aussi un bon moyen d'étudier les qualités et les défauts de la stratégie.

Au cours du développement, nous avons amélioré le rapport décrit dans les spécifications, en interface web. Cela nous permet d'avoir une interaction avec les résultats de la stratégie.

Une première page liste les stratégies exécutées par l'utilisateur en cours. Une fois une stratégie sélectionnée, on obtient un tableau contenant toutes les exécutions de la stratégie en fonction de ces paramètres. Ce tableau triable par colonne nous permet de prendre connaissance de la stratégie ayant la meilleure profitabilité où le ratio de perte minimum. En sélectionnant un de ces résultats d'exécution, nous arrivons sur une page nous donnant le détail d'exécution complet et un graphique interactif permettant de visualiser les différentes zones d'achats et de ventes.

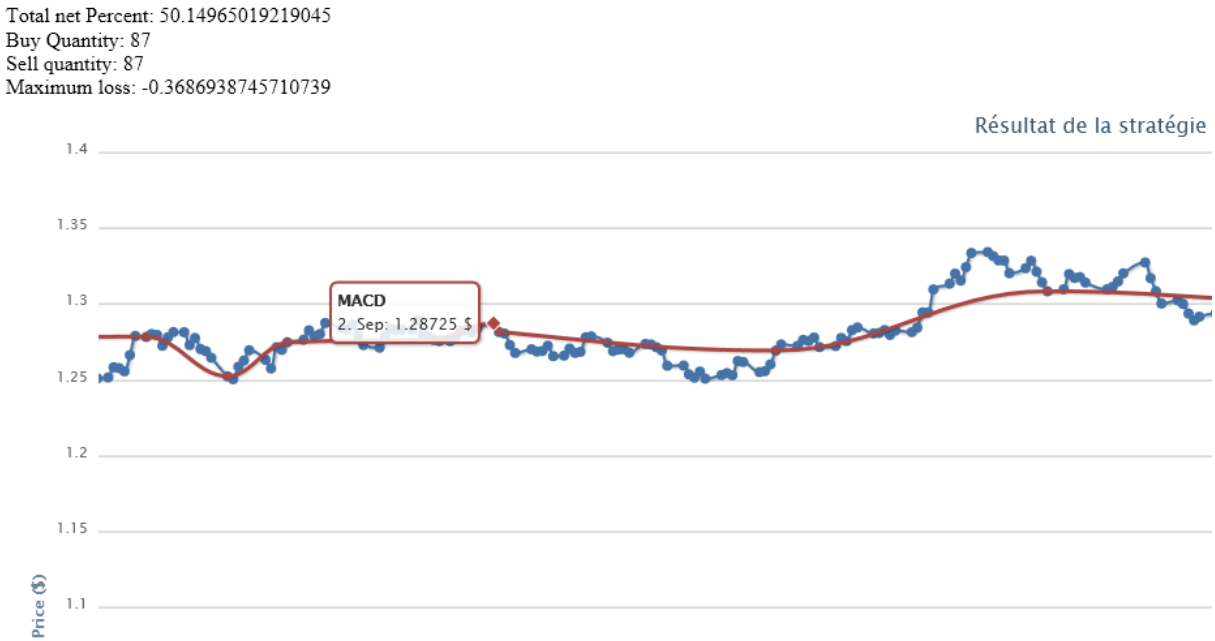


Figure 43 - Exemple de rapport interactif d'une stratégie

3.3.2. RÉSEAU DE NEURONES

Pour la réalisation des tests du réseau de neurones il faudra que nous mettions en place différents jeux de données. Comme décrit dans les spécifications techniques, il nous faudra comme jeu de test pour le réseau de neurones :

- Des cours réels en entrée du réseau.
- Des cours représentant l'idéal que l'on souhaite en sortie du réseau de neurones.
- Des cours réels d'évaluation différents des cours en entrée.

Pour réaliser nos tests, nous allons longuement entraîner notre réseau avec beaucoup de jeux de données pour peu de prédictions. Ainsi nous arriverons dans un premier temps à évaluer les paramétrages du réseau.

Nous prendrons comme écart de temps la journée. Les cours d'un jour à l'autre sont plus stables et incluent moins de bruit que ceux d'une seconde à l'autre par exemple.

Ainsi nous allons essayer de prédire 5 jours consécutifs, en donnant 10 ans de cours journaliers. Notre jeu d'entrée sera composé des cours au jour le jour du 1 mars 2002 au 1 mars 2012 soit 2729 jours de données.

Notre prédiction étant sur 5 jours, la couche de sortie du réseau de neurones contiendra 5 neurones. Le nombre de neurones de la couche d'entrée et de la couche cachée sera testé progressivement en fonction des résultats. Nous commencerons avec 5 neurones dans la couche d'entrée et 2 neurones dans la couche cachée.

Concernant le temps d'entraînement, nous avons décidé de comptabiliser l'entraînement en *itération* et non pas en minutes comme il est fait habituellement. L'itération permet de faire une comparaison entre deux entraînements à valeur égale.

Voici les 6 meilleurs résultats au niveau de la moyenne absolue de l'erreur d'un premier test :

Nombre d'itération	Neurones en entrée	Neurones cachées	Moyenne absolue de l'erreur	Symétrie directionnelle
18	20	3	0.000014640622312503963	75
13	20	3	0.000014640622312515501	60
15	20	2	0.000014640622312522632	55
14	20	2	0.000014640622312713634	70
16	20	2	0.00001464062231389871	85

Ce test a été lancé sur différents réseaux de neurones avec :

- Une couche en entrée avec des neurones allant de 5 à 30 neurones.
- Une couche cachée avec des neurones allant de 2 à 100 neurones.

- Une couche de sortie avec 5 neurones pour prédire nos 5 jours de cours de bourse.
- Un nombre d'itération allant de 1 à 100.

Voici un graphique représentant les 100 meilleurs résultats classés par moyenne absolue de l'erreur :

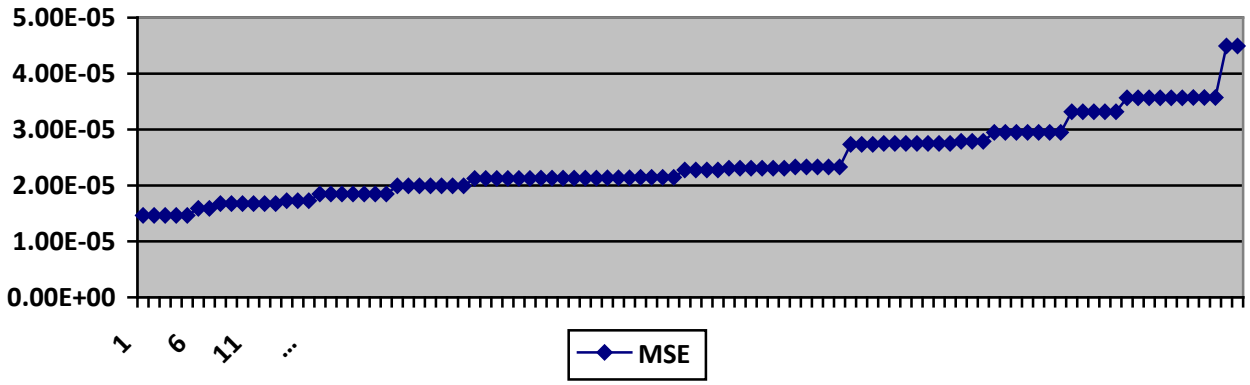


Figure 44 - Graphique des 100 meilleures moyennes absolues (meilleure à gauche)

L'on peut maintenant comparer ce graphique avec la symétrie directionnelle de ces 100 meilleurs résultats :

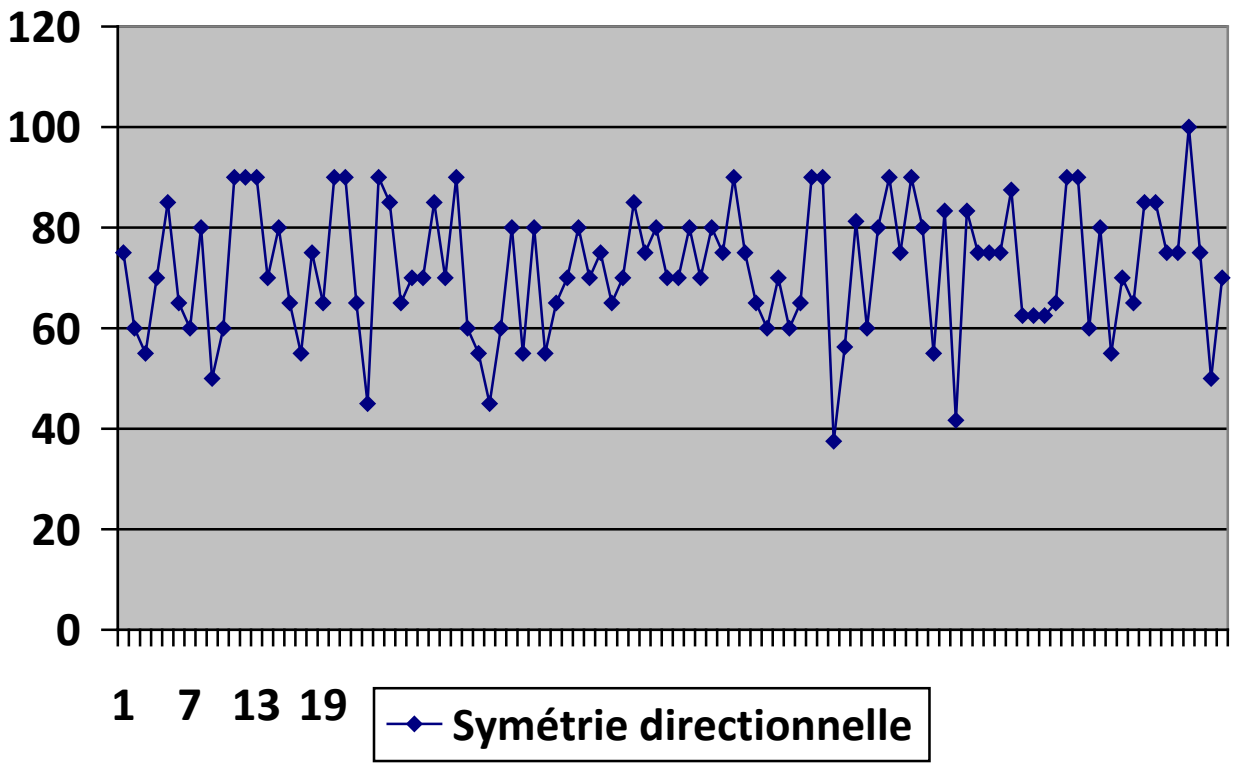


Figure 45 - Graphique représentant la symétrie directionnelle des 100 meilleurs résultats

Nous pouvons constater que la symétrie directionnelle n'est pas corrélée avec la moyenne absolue de l'erreur. En d'autres termes, quand la prédiction faite par le réseau de neurones est de plus en plus proche du modèle attendu, nous ne pouvons pas prévoir une amélioration ou une détérioration de la symétrie directionnelle, elle reste très aléatoire.

Toujours sur ces 100 meilleurs résultats voici un graphique représentant le nombre de neurones en entrée et dans la couche cachée :

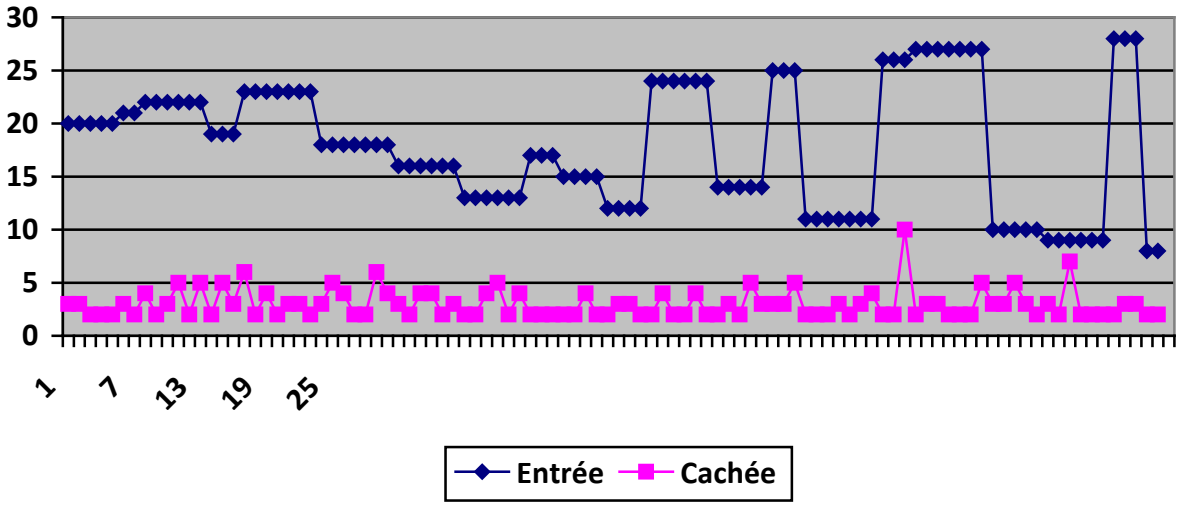


Figure 46 - Graphique représentant la quantité de neurones de la couche d'entrée et de la couche cachée des 100 meilleurs résultats

Nous pouvons constater sur ce graphique que le nombre de neurones pour la couche d'entrée tend entre 20 à 23 neurones, ce qui semble être l'idéal. Concernant la couche cachée le nombre de neurones est dans la fourchette de 2 à 6.

Les réseaux de neurones atteignent rapidement le surentraînement où le résultat des prédictions tend de plus en plus vers une ligne droite.

3.3.3. TRANSVERSE

Le module de sauvegarde est la partie qui va gérer la persistance des données et va agir comme cache pour l'accès aux données.

Ce module va être évalué à partir de :

- Sa vitesse d'accès aux données.
- Sa vitesse de sauvegarde des données.

Lors de la mise en route du système, nous chargeons directement les données les plus couramment utilisées en mémoire. Ceci a pour effet d'alourdir l'application mais de faciliter son accès aux données. Ainsi lors de l'arrivée de nouvelles données sur le bus de l'interface le module de sauvegarde récupère le nouveau cours. Ce dernier va être

sauvegardé en base. Cette sauvegarde nécessite de faire une insertion en base, donc une écriture sur le disque et une mise à jour d'index de la table. Par exemple une table contenant les cours de bourse maintient un identifiant numérique pour tous les enregistrements et un index sur la date du cours afin de le retrouver plus rapidement. Ce sont autant d'écritures et de mises à jour qui ralentissent le système lors de l'arrivée d'un nouveau cours.

Donc en plus de cette sauvegarde en base, le module va mettre à jour le cache afin d'accélérer l'accès aux données. Ceci permettra d'éviter la lecture de données en base au cours de l'exécution de l'application.

3.4. AMÉLIORATION

Nous allons donner ici les améliorations qu'il faudrait apporter au prototype afin qu'il devienne d'une part un système fini, et d'autre part qu'il soit exploitable dans un milieu de production.

3.4.1. AIDE À L'ANALYSE

Nous allons détailler les points à améliorer sur le module d'aide à l'analyse suivant l'analyse et les tests effectués dans la partie « Validation et performance ».

Voici les quatre points que nous devons améliorer :

- Compléter les fonctions des librairies.
- Accélérer le chargement des données.
- Accélérer le temps d'exécution.
- Compléter le rapport d'exécution.

Notre premier point d'amélioration est de compléter les librairies de fonctions. Celles-ci doivent être complétées au fur et à mesure des besoins mais peut aussi être améliorées en partant des librairies financières existantes tels que la librairie C++ : QuantLIB.

Le deuxième point concerne l'accélération de chargement des données. Le premier problème est que de toute manière les données nécessaires sont conséquentes. Cet accès

aux données peut toutefois être amélioré en ne chargeant dans un premier temps que le strict minimum. Par exemple si la stratégie n'utilise pas le prix d'ouverture du cours de bourse ou le prix le plus haut d'un cours de bourse sur une période, alors il n'est pas nécessaire de le charger.

Un autre axe d'amélioration possible, est le chargement séquentiel des données. Il suffit d'accéder aux données en base au fur et à mesure de leurs utilisations. Par exemple si la stratégie opère sur un million de cours comme c'est le cas pour une période de 10 ans avec un espace de temps d'une minute, nous pourrions charger les données par paquet de 10 000 cours au fur et à mesure des besoins.

Le troisième point concerne le temps d'exécution. Le module d'analyse a été développé de façon à exécuter une stratégie. Il faut le faire évoluer vers un système *multithreading* en prenant en compte le nombre de processeurs disponibles (voir le nombre de cœurs disponibles comme c'est le cas des processeurs actuellement) et en lançant pour chacun d'entre eux une stratégie à exécuter. Ainsi les exécutions seront réalisées en parallèle et permettront de réduire le temps global.

Le quatrième et dernier point concerne la qualité du rapport d'exécution. Il faut d'une façon générale l'améliorer afin qu'il permette d'analyser le résultat et de prendre des décisions concernant la conception de la stratégie. Pour cela, nous le ferons évoluer au fil du temps en fonction de l'expression des besoins par les utilisateurs et en fonction des catégories de stratégies utilisées.

3.4.2. RÉSEAU DE NEURONES

Bien que le réseau de neurones soit un système prometteur, les résultats que nous avons obtenus lors de la partie test sont relativement peu convaincants. Ceci se comprend par la nature de problèmes que peut résoudre un réseau de neurones : il peut résoudre des problèmes non linéaires.

Les cours de bourse évoluent et les méthodes d'investissement aussi. Rechercher un cours identique dans l'historique permet de retrouver un comportement identique. Mais un trader qui a investi en 2003 n'investit pas de la même façon en 2012. C'est pourquoi il est conseillé de coupler le réseau de neurones avec un algorithme génétique. Les algorithmes génétiques sont inspirés d'un mécanisme de sélection naturelle où les individus les plus forts sont les gagnants dans un environnement compétitif. En l'appliquant à notre cas, nous pourrons créer sous cet algorithme un nombre d'individus

représentant chacun un trader. Chaque trader évoluant au fil du temps en prenant des positions sur le cours plus ou moins profitable. L'algorithme génétique sélectionnerait simplement les plus performants d'entre eux. Puis il créerait de nouveaux traders en se basant sur les caractéristiques des anciens. Ainsi, au fur et à mesure de l'évolution des cours de bourse, l'algorithme aurait sélectionné les meilleurs traders.

Ceci est une des possibles évolutions du système du réseau de neurones. Elle est présentée et détaillée dans la thèse d'E. Kalyvas (2001).

D'autre part, M. Zhang (1996) présente une nouvelle typologie de réseau de neurones consacrée à la finance : les réseaux de neurones d'ordre élevé (en anglais « High Order Neural Network », HONN). Cette nouvelle typologie de réseaux apporterait un rendement supérieur de 8% par rapport à un réseau de type MLP.

3.5. AVENIR ET INDUSTRIALISATION DU SYSTÈME

La composition de notre système est relativement complète. Chaque fonction a été correctement découpée au sein de module n'ayant aucune tâche en doublon avec d'autres modules. Pour la rendre entièrement industrialisable, nous devrions premièrement réaliser des tests sur des machines professionnelles de type serveurs afin d'avoir une réelle estimation de la charge que le système peut supporter. En effet, il faudrait faire évoluer l'offre de système de trading vers une plateforme (machines et logiciels) plutôt qu'une simple application. Il faudrait pour cela réaliser une étude avec les différentes machines nécessaires afin de pouvoir supporter la charge. Le constructeur de matériel informatique Cisco (2012) propose un schéma pour la gestion haute performance du trading. Nous pouvons constater qu'une telle plateforme représente plusieurs centaines de milliers d'euros d'investissements. Ainsi, il faudrait que notre offre se complète avec une offre matérielle afin d'apporter au client une offre *package* tout compris.

Cela nécessitant de gros investissements, il faudrait penser à découper cette offre et ainsi atteindre des objectifs étape par étape. Par exemple, nous pourrions attaquer le marché des gestionnaires de protocoles FIX correspondant à notre module *Interface* avec une offre matérielle qui serait bien plus abordable que le système dans son ensemble.

Une autre possibilité envisageable, moins concurrentielle est le marché de l'élaboration de stratégies par les particuliers. Même si le marché reste très restreint aux personnes

sachant développer en java, il peut devenir très intéressant de vendre un service de courtier et ainsi facturer à la transaction comme peut le faire un courtier classique.

Mais à l'heure actuelle, nous devons compléter ce système avec un maximum de fonctionnalités.

4. ANNEXE

4.1. EXEMPLE DE PRÉDICTION

Dans cette annexe est présenté un exemple de prédiction détaillé en fournissant le type de réseau de neurones, les paramètres d'initialisation du réseau de neurones, les jeux de tests ainsi que les résultats. Nous donnons l'évolution de l'état du réseau de neurones au fur et à mesure de son apprentissage et de ses prédictions.

4.1.1. LE RÉSEAU DE NEURONES

Le réseau de neurones utilisés ici est le logiciel Encog de Heaton Research (www.heatonresearch.com/encog). Il a l'avantage d'être Open Source et d'être entièrement codé dans le langage java.

Ce logiciel fournit différentes typologies de réseau de neurones (ADALINE, machine de Boltzmann, Hopfield, …). Comme détaillé tout au long de ce mémoire, nous utilisons une typologie multicoches à rétro propagation (en anglais « feedforward »).

Les analyses présentées dans le chapitre 3.3, nous permettent de paramétrer le réseau de neurones avec 3 couches :

- 1 couche d'entrée
- 1 couche cachée
- 1 couche de sortie

Le nombre de neurones de chaque couche est paramétré toujours grâce aux précédentes analyses :

- Couche d'entrée : 20
- Couche cachée : 3
- Couche de sortie : 10

Ces paramètres déterminent le nombre de cours de bourse (20, nombre de neurones de la couche d'entrée) que nous fournirons au réseau de neurones afin qu'il génère une prédiction et le nombre de cours de bourse prédit (10, nombre de neurones de la couches de sorties).

Voici la figure représentant cette typologie avec le nombre de couches et le nombre de neurones pour chaque couche (le nombre de flèches reliant les neurones entre eux ont été simplifiés) :

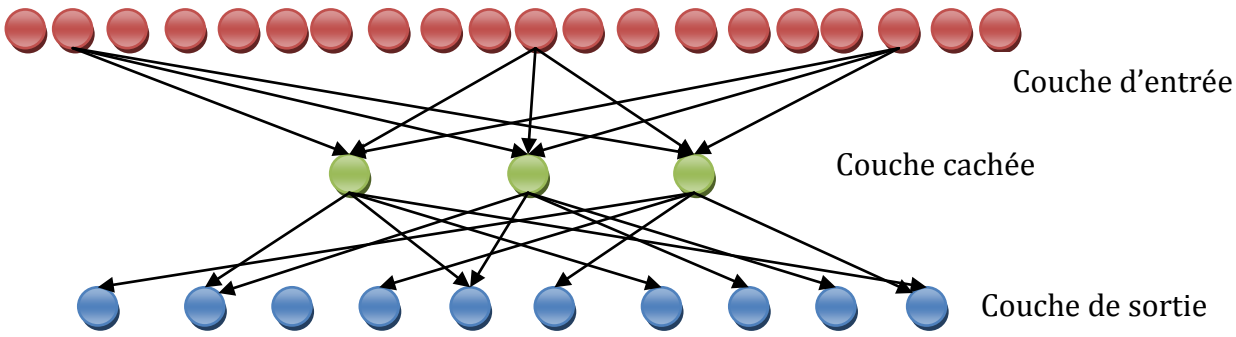


Figure 47 - Schéma du réseau de neurones testé

4.1.2. INITIALISATION

Le réseau de neurones est initialisé de manière aléatoire. C'est-à-dire que lors de son initialisation, il va générer les poids pour chacun de ces neurones en utilisant la méthode de *Nguyen-Widrow* dans une fourchette de -1 à 1.

Voici un tableau représentant les poids de chaque neurone de la couche cachée à la couche de sortie après initialisation:

Neurone source	Neurone destination	Poids
1	1	0.229403
1	2	-0.34857
1	3	-0.23568
1	4	0.316063
1	5	0.005251

1	6	-0.37214
1	7	-0.31207
1	8	-0.08052
1	9	0.195278
1	10	-0.44785
2	1	-0.57792
2	2	0.375853
2	3	0.512664
2	4	0.368442
2	5	0.252495
2	6	-0.40186
2	7	-0.37586
2	8	-0.08343
2	9	-0.33938
2	10	0.095646
3	1	0.394461
3	2	-0.23909
3	3	-0.28282
3	4	-0.0696
3	5	0.451465
3	6	-0.28108
3	7	0.155783
3	8	-0.54706
3	9	0.17757
3	10	-0.01611

4.1.3. JEUX DE DONNÉES

La deuxième étape d'initialisation du réseau de neurones passe par les phases d'apprentissage et de test. Ces deux phases permettent d'entrainer le réseau de neurones à fournir des prédictions, puis d'évaluer ses prédictions.

Pour cela, nous lui fournirons un jeu de données pour l'apprentissage, puis nous nous servirons d'un deuxième jeu de données pour le tester.

Le jeu de données d'apprentissage comprend dix ans de cours de bourse des devises euro contre dollar (cours au jour le jour). Ses cours sont pris depuis l'an 2000 jusqu'en 2010. Voici le graphique représentant ces données sur une période de 10 ans :

Figure 48 - Cours de bourse de l'euro/dollar depuis 10 ans

Voici un exemple de ces cours sur une période d'un mois (Janvier 2009) :

Date	Prix de l'euro/dollar (en dollar)	Différence en %
2009-01-01 23:00:00	1.38537	
2009-01-04 23:00:00	1.36376	-0.01585
2009-01-05 23:00:00	1.35386	-0.00731
2009-01-06 23:00:00	1.3649	0.008089
2009-01-07 23:00:00	1.37047	0.004064
2009-01-08 23:00:00	1.34313	-0.02036
2009-01-11 23:00:00	1.33644	-0.00501
2009-01-12 23:00:00	1.31865	-0.01349
2009-01-13 23:00:00	1.31937	0.000546
2009-01-14 23:00:00	1.31122	-0.00622

2009-01-15 23:00:00	1.32912	0.013468
2009-01-18 23:00:00	1.30709	-0.01685
2009-01-19 23:00:00	1.29075	-0.01266
2009-01-20 23:00:00	1.30256	0.009067
2009-01-21 23:00:00	1.30037	-0.00168
2009-01-22 23:00:00	1.29869	-0.00129
2009-01-25 23:00:00	1.31922	0.015562
2009-01-26 23:00:00	1.31632	-0.0022
2009-01-27 23:00:00	1.31681	0.000372
2009-01-28 23:00:00	1.29567	-0.01632
2009-01-29 23:00:00	1.28063	-0.01174

Le jeu de données de test des prédictions sont les cours des mêmes devises sur l'année suivant ses cours, c'est-à-dire l'année 2011 :

Figure 49 - Cours de bourse de l'euro/dollar de l'année 2011

Voici un exemple de ces cours sur une période d'un mois (Janvier 2011) :

Date	Prix de l'euro/dollar (en dollar)	Différence en %
2011-01-02 23:00:00	1.3362	
2011-01-03 23:00:00	1.33089	-0.00399
2011-01-04 23:00:00	1.31512	-0.01199
2011-01-05 23:00:00	1.30044	-0.01129
2011-01-06 23:00:00	1.29125	-0.00712
2011-01-09 23:00:00	1.29528	0.003111
2011-01-10 23:00:00	1.29756	0.001757
2011-01-11 23:00:00	1.31319	0.011902
2011-01-12 23:00:00	1.33645	0.017404
2011-01-13 23:00:00	1.33749	0.000778
2011-01-16 23:00:00	1.32946	-0.00604
2011-01-17 23:00:00	1.33883	0.006999
2011-01-18 23:00:00	1.34746	0.006405
2011-01-19 23:00:00	1.34735	-8.2E-05
2011-01-20 23:00:00	1.36162	0.01048
2011-01-23 23:00:00	1.36394	0.001701
2011-01-24 23:00:00	1.3683	0.003186
2011-01-25 23:00:00	1.37131	0.002195
2011-01-26 23:00:00	1.37348	0.00158
2011-01-27 23:00:00	1.36099	-0.00918
2011-01-30 23:00:00	1.36936	0.006112
2011-01-31 23:00:00	1.38314	0.009963

4.1.4. APPRENTISSAGE

Nous détaillons ici la phase d'apprentissage et plus précisément le changement des poids. Nous connaissons déjà la valeur des poids à l'initialisation. Nous allons préciser maintenant au réseau de neurones que nous souhaitons qu'il itère 1000 fois sur chacune des données afin d'ajuster ces poids sans tomber dans le surentrainement.

Neurone source	Neurone destination	Poids	Poids après 1ére itération	Différences
1	1	0.229403	0.129402982	0.1
1	2	-0.34857	-0.248568904	-0.1
1	3	-0.23568	-0.135684565	-0.1
1	4	0.316063	0.416063016	-0.1
1	5	0.005251	-0.094748997	0.1
1	6	-0.37214	-0.27214083	-0.1
1	7	-0.31207	-0.212072598	-0.1
1	8	-0.08052	0.01947979	-0.1
1	9	0.195278	0.295278254	-0.1
1	10	-0.44785	-0.347850655	-0.1
2	1	-0.57792	-0.477920264	-0.1
2	2	0.375853	0.275852879	0.1
2	3	0.512664	0.412664303	0.1
2	4	0.368442	0.268441794	0.1
2	5	0.252495	0.352495013	-0.1
2	6	-0.40186	-0.501858301	0.1

2	7	-0.37586	-0.475856675	0.1
2	8	-0.08343	-0.183433934	0.1
2	9	-0.33938	-0.439380943	0.1
2	10	0.095646	-0.004354273	0.1
3	1	0.394461	0.494460958	-0.1
3	2	-0.23909	-0.339089456	0.1
3	3	-0.28282	-0.382816652	0.1
3	4	-0.0696	-0.169595933	0.1
3	5	0.451465	0.551464553	-0.1
3	6	-0.28108	-0.381082903	0.1
3	7	0.155783	0.055782931	0.1
3	8	-0.54706	-0.647064976	0.1
3	9	0.17757	0.077570302	0.1
3	10	-0.01611	-0.11611402	0.1

La quatrième colonne du tableau nous indique le réajustement des poids qu'il y a eu pour cette première itération. Nous remarquons que le réajustement est particulièrement important par rapport à la valeur des poids.

Si nous comparons cette itération avec les suivantes, nous obtenons le tableau suivant :

Poids après 1ére itération	**Poids après 2éme itération**	**Différences**	**Poids après 3éme itération**	**Différences**
0.129402982	0.009403	0.12	-0.134597	0.144
-0.248568904	-0.348569	0.1	-0.398569	0.05
-0.135684565	-0.235685	0.1	-0.285685	0.05

0.416063016	0.316063	0.1	0.266063	0.05
-0.094748997	0.005251	-0.1	0.055251	-0.05
-0.27214083	-0.372141	0.1	-0.322141	-0.05
-0.212072598	-0.312073	0.1	-0.362073	0.05
0.01947979	-0.08052	0.1	-0.13052	0.05
0.295278254	0.1952783	0.1	0.1452783	0.05
-0.347850655	-0.447851	0.1	-0.497851	0.05
-0.477920264	-0.57792	0.1	-0.62792	0.05
0.275852879	0.1558529	0.12	0.0118529	0.144
0.412664303	0.2926643	0.12	0.1486643	0.144
0.268441794	0.1484418	0.12	0.0044418	0.144
0.352495013	0.472495	-0.12	0.616495	-0.144
-0.501858301	-0.621858	0.12	-0.501858	-0.12
-0.475856675	-0.595857	0.12	-0.739857	0.144
-0.183433934	-0.303434	0.12	-0.447434	0.144
-0.439380943	-0.559381	0.12	-0.703381	0.144
-0.004354273	-0.124354	0.12	-0.268354	0.144
0.494460958	0.394461	0.1	0.344461	0.05
-0.339089456	-0.459089	0.12	-0.603089	0.144
-0.382816652	-0.502817	0.12	-0.646817	0.144
-0.169595933	-0.289596	0.12	-0.433596	0.144
0.551464553	0.6714646	-0.12	0.8154646	-0.144
-0.381082903	-0.501083	0.12	-0.381083	-0.12

0.055782931	-0.064217	0.12	-0.208217	0.144
-0.647064976	-0.767065	0.12	-0.911065	0.144
0.077570302	-0.04243	0.12	-0.18643	0.144
-0.11611402	-0.236114	0.12	-0.380114	0.144

Nous pouvons constater qu'au fur et à mesure des itérations, l'algorithme de rétro propagation (fonctionnement détaillé ch. 3.2.4) ajuste les poids de façon de plus en plus fine.

4.1.5. TEST

Nous passons maintenant à la phase de test qui va nous permettre d'apprécier la phase d'apprentissage du réseau de neurones.

En lui fournissant le jeu de données affiché précédemment (cours de bourse de janvier 2011), nous allons pouvoir présenter les prédictions :

La première prédiction comprend en entrée du réseau de neurones les 20 premiers cours de bourse (en pourcentage de différence avec le cours précédent). À l'idéal, la prédiction devrait nous donner les 10 jours suivant de cours (toujours en pourcentage) :

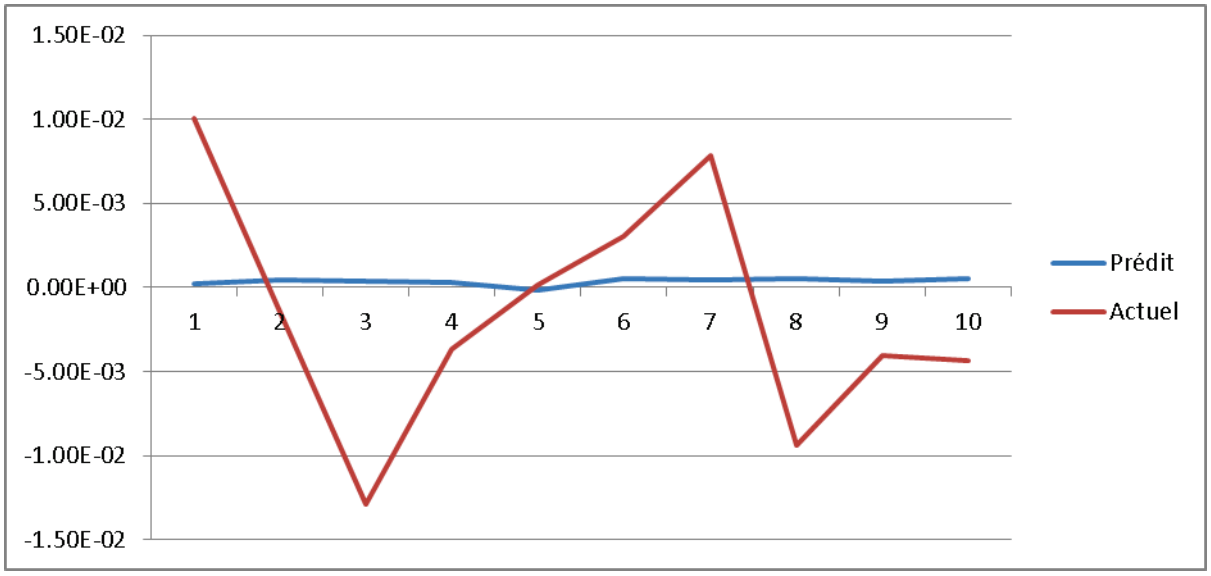
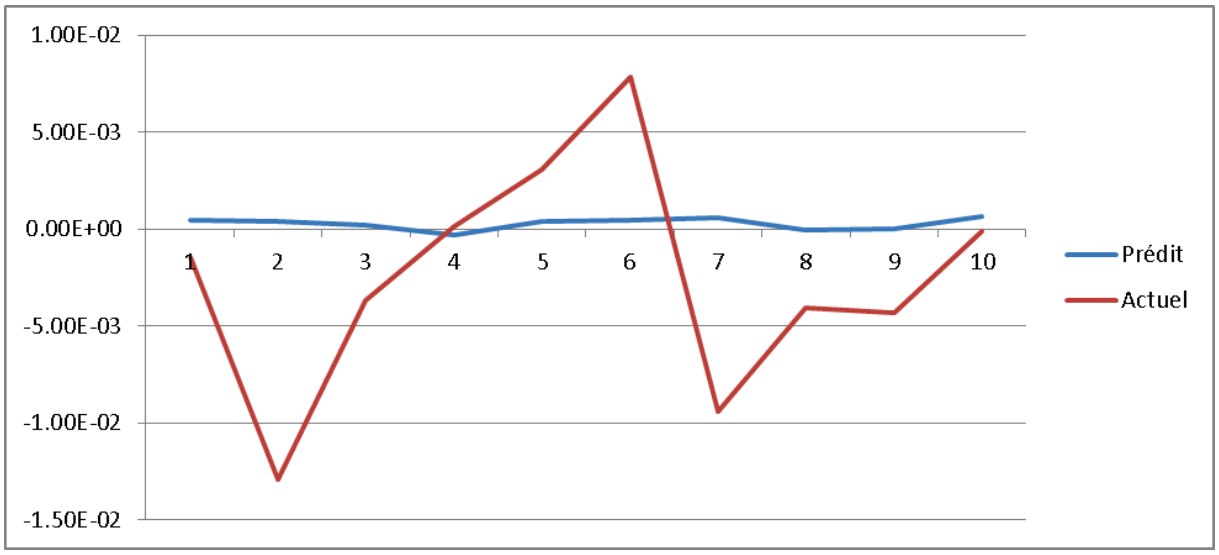


Figure 50 - Prédiction d'un cours de l'euro dollar sur 10 jours

Ce graphe nous permet de constater que la prédiction est relativement éloignée du cours idéal. En reprenant les indicateurs de performance détaillée dans le chapitre *Validation et performance*, nous avons une symétrie directionnelle de 22 sur 100 (ce qui est très faible) et une moyenne de l'erreur absolue de -0.0057.

Sur un deuxième exemple, toujours en fournissant les données du mois de janvier 2011 décalé d'un jour par rapport à l'exemple précédent, nous obtenons le graphique suivant :



Le cours prédit est toujours très éloigné du cours actuel, cependant, la symétrie directionnelle est 44 sur 100, ce qui prouve d'une nette amélioration de la prédiction sans pour autant être fiable.

5. CONCLUSION

La spécification de notre système de trading et le développement de celui-ci, nous permet d'avoir une bonne vision de la complexité des systèmes *front-office* que peuvent utiliser les institutions financières. En ayant mis au point une stratégie de tests et de réseau de neurones, nous prouvons la bonne conception de notre système à intégrer des modules hétérogènes ayant différents fonctionnements. Nous sommes certains de pouvoir continuer à faire évoluer les besoins et les spécifications du système afin de les rendre complets et totalement opérationnels.

Ce mémoire et notamment le prototype ont pu être réalisés grâce à des intervenants du domaine financier, extérieurs au projet qui ont pu m'aider à éclaircir d'importantes

zones d'ombre. De plus, j'ai pu m'appuyer sur de solides connaissances acquises au cours de mes expériences professionnelles notamment à la banque d'investissement de la Société Générale en tant que *Business Analyst*, position qui m'a permise de beaucoup apprendre sur les risques du trading et leurs méthodes de calculs. J'ai, de plus, travaillé chez Natixis toujours en tant que *Business Analyst* où nous évaluions la valeur des entreprises ce qui m'a beaucoup aidé dans la spécification des besoins particulièrement des stratégies. Ce projet a été réalisé dans un contexte et avec des moyens personnels tout en ayant comme objectif de commercialiser une solution finale. Les différents segments de marchés que nous avons évoqués et plus particulièrement les méthodes que nous pourrions utiliser pour vendre notre produit, nous permettraient de nous introduire sur le marché du logiciel financier en commençant petit et en étudiant les besoins personnalisés des clients.

Plus concrètement, j'ai établi un calendrier pour faire évoluer le prototype en système tout en suivant l'objectif de mettre en place un système de trading automatique pour particulier averti. Les grandes étapes de ce calendrier sont :

- Compléter les spécifications pour définir exactement les fonctions de chaque partie du système (Mai 2012).
- Développer chaque module suivant ces nouvelles spécifications et partant des développements réalisés sur le prototype (Juin/Juillet/Aout/Septembre 2012).
- Test et mise en ligne du système (Octobre 2012)
- Recherche d'un fournisseur des cours de bourse (Novembre 2012)
- Développement et test de la connexion avec ce fournisseur (Décembre 2012)
- Tests généraux avant lancement (Janvier 2013)
- Lancement (Février 2013)

Sur la base de ce calendrier, on peut estimer mettre en ligne une première version française du système au premier trimestre 2013.

D'un point de vue personnel, la rédaction de ce mémoire m'a énormément apporté. Il a permis de réaliser ma première étude approfondie sur un sujet ainsi qu'à la structurer. Le point particulièrement sensible a été sans aucun doute le réseau de neurones. Sa compréhension et sa spécification ont été difficiles à appréhender. Toutefois, une fois que j'ai pu trouver de la documentation de qualité sur laquelle baser mon étude,

l'écriture du mémoire a été grandement facilitée. Il est clair qu'à posteriori, et si c'était à refaire, je ferai différemment : je choisirais de concentrer le mémoire uniquement sur la prédiction de cours ou uniquement sur les réseaux de neurones appliqués à la finance. J'ai entrepris ce mémoire qui aborde un sujet vaste et peu documenté en général. Toutefois, j'en sors indéniablement avec de meilleures compétences d'études et suis maintenant persuadé qu'il me permettra d'atteindre les objectifs voulus.

6. BIBLIOGRAPHIE

1. (en) Irène Aldrige, Introduction. In *High-Frequency Trading - A Practical Guide to Algorithmic Strategies and Trading Systems*. Chapitre 1. Wiley, 2010.

2. (en) Irène Aldrige, Overview of the Business of High-Frequency Trading. In *High-Frequency Trading - A Practical Guide to Algorithmic Strategies and Trading Systems*. Chapitre 3. Wiley, 2010.

3. (en) Irène Aldrige, Evaluating Performance of High-Frequency Strategies. In *High-Frequency Trading - A Practical Guide to Algorithmic Strategies and Trading Systems*. Chapitre 5. Wiley, 2010.

4. (en) Stuart Russel, Peter Norvig. *Artificial Intelligence: A Modern Approach.* Pearson, 2010

5. (en) Toshinori Munakata, Neural Networks: Fundamentals and the Backpropagation Model. *Fundamentals of the New Artificial Intelligence – Neural, Evolutionary, Fuzzy and more.* Chapitre 2. Springer, 2008.

6. (en) Minsky et Papert. In *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1969.

7. (en) Gregor Hohpe, Bobby Woolf, Messaging Systems. In *Enterprise Integration Patterns*. Chapitre 3, Wesley, 2011.

8. (en) James A. Freeman et David M. Skapura, Backpropagation. In *Neural Networks Algorithms, Applications, and Programming Techniques.* Chapitre 3, Wesley, 1991

9. (en) Ben Kröse et Patrick van der Smagt, Back-Propagation. In *An introduction to Neural Networks*. Chapitre 4, The University of Amsterdam, 1996.

10. (en) M. Zhang, Automatically Identifying Predictor Variables for Stock Return Prediction. In *Artificial Higher Order Neural Networks for Economics and Business*. Chapitre 3, ISR, 1996

11. (en) Richard P. Lippmann, Single Layer Perceptron. In *An Introduction to Computing with Neural Nets*. IEEE, 1987.

12. (en) K.Mehrotra, C. K. Mohan et S. Ranka, Introduction. In *Elements of Artificial Neural Networks*. Chapitre 1, MIT Press - 1996

13. (en) Laurene Fausett, Introduction. In *Fundamentals of Neural Networks*. Chapitre 1, Prentice Hall, 1993.

14. (en) Mendel, J. M. and McLaren. In *Adaptive, Learning and Pattern Recognition Systems: Theory and Applications*. Academic Press, New York, 1970.

15. (en) Simon Haykin, Learning Processes. In *Neural Network – A Comprehensive Foundation*. Chapitre 2, Pearson, 2005.

16. (en) W. Remus, M. O'connor, Neural Networks For Time Series Forecasting. In *Principles of Forecasting: A Handbook for Researchers and Practitioners*. Kluwer Academic Publishers, 2001.

17. (en) M. Zhang, Foreign Exchange Rate Forecasting Using Higher Order Flexible Neural Tree. In *Artificial Higher Order Neural Networks for Economics and Business*. Chapitre 5, ISR, 1996

18. (en) Efstathios Kalyvas, Models .*Using neural networks and genetic algorithms to predict stock market returns*. Chapitre 4, Université de Manchester, 2001

7. BIBLIOGRAPHIE INTERNET

1. (en) John Mc Carthy, What is Artificial Intelligence ? In *http://www-formal.stanford.edu/jmc/whatisai/node1.html*, mars 2012

2. (en) John Mc Carthy, What is Intelligence ? In *http://www-formal.stanford.edu/jmc/whatisai/node1.html*, mars 2012

3. (en) FIX Organization, Background. In *http://fixprotocol.org/what-is-fix.shtml*. mars 2012

4. NYSE Euronext. In *http://www.nyxdata.com/*. mars 2012

5. Cisco. In http://www.cisco.com/web/strategy/docs/finance/HPT_Poster_hires_060311.pdf. Mars 2012