From Prediction to Classification – The Application of Pattern Recognition Theory to Stock Price Movements Analysis

Fabian Simillion
Inforge - HEC - BFSH1
University of Lausanne
CH-1015 Lausanne-Dorigny
Switzerland
Phone : + 41 21 6923426
Fax : + 41 21 6923405
attn. F.Simillion
e-Mail: fabian.simillion@hec.unil.ch

Jean Hennebert
Chaire de Circuits et Systèmes - DE
Swiss Federal Institute of Technology (EPFL)
CH-1015 Lausanne
Switzerland
Phone : + 41 21 6934687
Fax : + 41 21 6936700
attn. J.Hennebert
e-Mail: jean.hennebert@circ.epfl.ch

Maia Wentland
Inforge - HEC - BFSH1
University of Lausanne
CH-1015 Lausanne-Dorigny
Switzerland
Phone : + 41 21 6923429
Fax : + 41 21 6923405
attn. M.Wentland
e-Mail: maia.wentland@hec.unil.ch

Abstract

The limited success of most prediction systems has proved that future stock prices are very difficult to predict. The purpose of this paper is to show that future prices do not have to be known to make successful investments and that anticipating movements (increases or decreases) of the price can be sufficient. Probabilistic classification systems based on pattern recognition theory appear to be a good way to reach this objective. Moreover, they include some other advantages, principally in terms of risk management. Results show satisfactory classification hit rates but a rather poor translation into financial gains. This paper tries to identify causes of this problem and proposes some ideas of solution.

Keywords : Parzen Windows, Artificial Neural Networks, Classification, Probabilities, Decision Support System

FROM PREDICTION TO CLASSIFICATION – THE APPLICATION OF PATTERN RECOGNITION THEORY TO STOCK PRICE MOVEMENTS ANALYSIS

Fabian Simillion / Inforge - HEC / University of Lausanne / Lausanne - Switzerland
e-Mail: fabian.simillion@hec.unil.ch

Jean Hennebert / Department of Electricity / CIRC - EPFL / Lausanne - Switzerland
e-Mail: jean.hennebert@circ.epfl.ch

Maia Wentland / Inforge - HEC / University of Lausanne / Lausanne - Switzerland
e-Mail: maia.wentland@hec.unil.ch

ABSTRACT

The limited success of most prediction systems has proved that future stock prices are very difficult to predict. The purpose of this paper is to show that future prices do not have to be known to make successful investments and that anticipating movements (increases or decreases) of the price can be sufficient. Probabilistic classification systems based on pattern recognition theory appear to be a good way to reach this objective. Moreover, they include some other advantages, principally in terms of risk management. Results show satisfactory classification hit rates but a rather poor translation into financial gains. This paper tries to identify causes of this problem and proposes some ideas of solution.

1. INTRODUCTION

When intending to invest in financial markets (the stock market for example), one is immediately confronted with a dilemma. *Either* to accept the "efficient-market theory", whose different versions have dominated market theory in recent years. *Or* to say that the theory may be "roughly" true, and that loopholes of inefficiency do exist and can be exploited.

The weakest form of the theory states that in a market that assimilates information efficiently it is impossible to predict the future price of securities on the basis of its past prices: a system based on past price analysis can not "beat the market". So, the solution is to follow financial experts' advice or to apply the "buy and hold" strategy, and to minimize the risk by using the Capital Asset Pricing Model (CAPM) or other similar models [Brealey and Myers, 1991].

Recent observations however reinforce the viewpoint against the efficient-market theory [related in The Economist, 1993]. It has been shown that the correlation between past and future returns was significantly different from zero and that some of the most popular chartist techniques work quite well. Studies have also outlined that events on the stock market are not produced by random processes but by processes that tend to go in runs. This reveals that *something* is causing the stock market to be predictable, or in other words, to have "memory".

It appears to be difficult however to outperform financial markets. Researchers have not awaited observations of imperfect efficiency to start building forecasting models. They have analyzed price movements in a lot of ways. They have made a lot of complicated calculations and sometimes found quite surprising explanations. But, as Brealey and Myers put it, « if many of these researchers have become famous, none have become rich... » [Brealey and Myers, op.cit., p.293].

Up to now, most methods have aimed at predicting future values of stock prices. The Box-Jenkins method is one of the most renowned in business and economics. It uses a systematic procedure to select an appropriate model from a rich family of models, namely, ARIMA models [Box and Jenkins, 1976]. The main problem with this method is that it uses linear models, whereas financial markets seem to be highly non-linear (meaning that same-sized causes can have different-sized effects, depending on the circumstances), therefore making the prediction not very reliable.

Thus, in order to exploit market memory, there is a need for using non-linear methods. Among them, connexionist systems have drawn considerable attention in recent years because of their learning abilities [Refenes, Azema-Barac and Treleaven, 1992]. Different network types and configurations have been tested in order to forecast future values of time series. Publicly available results have often been promising but are still insufficient to be applied with confidence in practice.

We have tried to look at this problem from an innovative point of view. Trying to predict directly future values is obviously a very difficult task. Moreover, even with good prediction results, there is a lack of methods able to associate confidence levels to predictions. To make successful investments, we believe that *anticipating future stock prices is not necessary* and that *concentrating on increases and decreases can be sufficient*. In other words, by replacing prediction with classification, our goal is to focus on a less complex problem in order to solve it better.

After showing why prediction and classification can be regarded as variations of the same approach, we will explain how we replaced the first by the latter. We will then briefly describe our system, some of its theoretical aspects, and the results obtained. We will conclude by discussing our future projects, grounded on what has already been achieved.

2. *FROM PREDICTION TO CLASSIFICATION*

Generally, a prediction system (linear or non-linear) realizes a function

$$\hat{x}_t = f\left(x_{t-1}, x_{t-2}, \ldots, x_{t-p}\right), \qquad [1]$$

where p is the order of the system and where $\hat{x}_t$ is the predicted value of x_t.

Building a prediction system requires first to discover the order of the system and then to determine both the nature and the parameters of the function[1]. Once p and the nature of the function are fixed, the parameters are generally tuned to minimize the mean square error (MSE criterion) on the sequence:

$$E = \sum_{t=1}^{T} \left(x_t - \hat{x}_t\right)^2 . \qquad [2]$$

Whatever the nature of the function f, the minimization of E is equivalent to a correct estimation of the conditional probability [Papoulis, 1965]:

[1] This paper does not discuss the order estimation problem that is still unsolved. Different methods dealing with this topic are presented in [Takens, 1980] and [Ris *et al.*, 1994].

$$P\left(x_t \middle| x_{t-1}, x_{t-2}, \ldots, x_{t-p}\right). \qquad [3]$$

The approach presented in this paper consists of classifying a sequence of prices (x_{t-1}, x_{t-2}, ..., x_{t-p}) into two classes (q_1, q_2) corresponding to a rising or a falling price in *t*. It can then be seen as a simplified version of equation [1]. Conversely, our two-classes approach can be extended towards a full prediction problem by multiplying the number of classes. If we increase the number of classes toward infinity, we can obtain the full prediction problem of equation [1].

3. Why Classification Instead Of Prediction?

We mentioned above that our goal is to focus on a less complex problem in order to solve it better. What do we mean with "less complex" and "solve better"?

Considering that a prediction can be seen as a classification into a infinity of categories, handling the problem of a two-category classification is certainly less complex than building a forecasting model.

Solve it better refers to getting more valuable information as output in order to invest more opportunely on the market. Obtaining probabilities of increase and decrease conditioned on a particular input instead of a "raw" estimation of a future value is a good example of valuable information.

A simple prediction system just says: « I predict that the next value will be X. », but does not indicate the probability attached to its prediction. No information is given about how risky the investment is, and therefore *risk-aversion* cannot be taken in consideration.

Using classification and probabilities allows more accurate actions. The main idea is that *probabilities provided by the system can be interpreted as measures of confidence* that the system gives itself to its outputs. On one hand, if both probabilities are close to 50%, it means that the system has already encountered similar situations that have led to increases as well as to decreases. So, even if one probability is slightly higher than the other, it could be better to "stay" (not invest or "disinvest"). On the other hand, if the probabilities are significantly different, it becomes interesting to "move" (buy or sell), with a confidence proportional to the difference between the two probabilities.

On financial markets, where it exists a risk in every investment, this advantage is crucial. It will allow investors to take this risk into consideration and to tune their investments according to their risk aversion.

The only disadvantage that we see with this approach is that it does not give any indication of the increase or decrease scale. Is it worth investing if the future increase is 0.1%? Of course not! Transaction costs would make the operation a failure. A solution could be to increase the number of categories (see *6. Future Works*).

Therefore, our goal is to make a probabilistic classification. To achieve this, we used pattern recognition systems, which are based on probabilities. For a given input pattern and for each category, they give one conditional probability that the pattern belongs to the category.

The objective is to classify a sequence of values ($\mathbf{x}_{t-1}$, $\mathbf{x}_{t-2}$, ... , $\mathbf{x}_{t-p}$) into two classes (q_1, q_2) corresponding to a rising or a falling tendency in *t* and performing a short term analysis of the price behavior. The input vector ($\mathbf{x}_{t-1}$, $\mathbf{x}_{t-2}$, ... , $\mathbf{x}_{t-p}$) will belong to q_1 if ($\mathbf{x}_t$-$\mathbf{x}_{t-1}$) $\geq$ 0 and to q_2 if ($\mathbf{x}_t$-$\mathbf{x}_{t-1}$) < 0. Doing that, we have to make the assumption that the distribution of the pattern ($\mathbf{x}_{t-1}$, $\mathbf{x}_{t-2}$, ... , $\mathbf{x}_{t-p}$) is not evolving in time (stationary), which can be criticized for a long term analysis. Recent researches however have put in evidence the existence of "pocket of predictability", that is, periods during which it is possible to derive some prediction abilities [related in The Economist, op.cit.]. In practice, it means that we will have to train our system over a period of time short enough to fall in a pocket of predictability and long enough to be able to come out with reliable statistics.

4. SYSTEM DESCRIPTION

As shown in *Figure 1*, the system is composed of different elements that exchange different types of information.

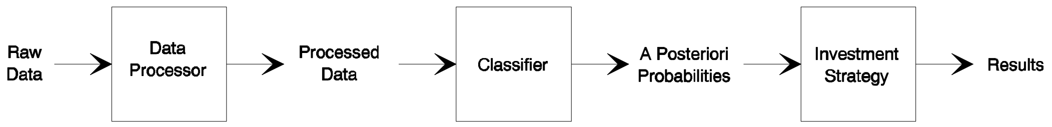


Figure 1 - Elements of the classification system.

4.1. RAW DATA

We have used daily stock price series of several major companies in Switzerland, namely, Nestlé, Ciba-Geigy, Sandoz, Zurich Insurances and Union Bank of Switzerland (UBS) covering the period between January'88 and June'94 inclusive. Though most series started before '88, we decided to drop pre-'88 data to avoid the learning of out-of-date behaviors. The chart in *Figure 2* shows the evolution of Nestlé's stock price during the chosen period.

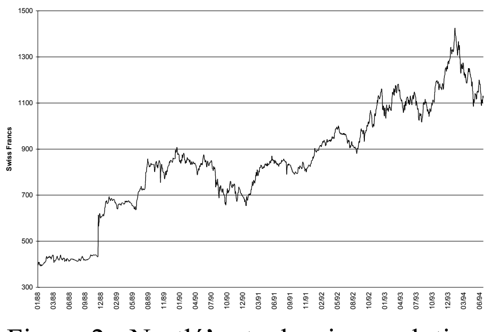


Figure 2 - Nestlé's stock price evolution from January'88 to June'94.

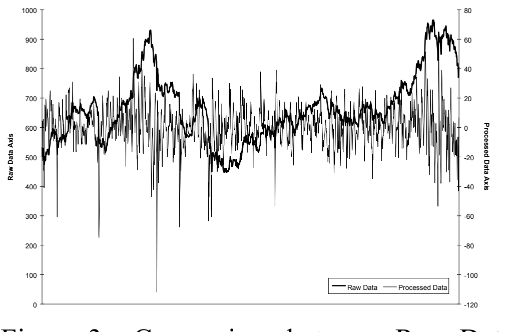


Figure 3 - Comparison between Raw Data and Processed Data for Ciba-Geigy's stock.

The first 70% of the data set were used for training (1185 days), the 30% left being reserved for the test (509 days).

4.2. DATA PROCESSOR

From *Figure 2*, we can conclude that stock prices follow trends on quite long periods. Since our objective is to study short-term prices fluctuation, we need to perform a preprocessing that takes away the long-term tendency. Preprocessing has been proved to improve the quality of outputs in many experiences, especially in cases where data follow trends or cycles [Kouam, Badran and Thiria, 1991].

We used a rudimentary non-weighted moving average filter of size 5 given by the following formula:

$$Y_t = X_t - \frac{\sum_{i=1}^{5} X_{t-i}}{5} \qquad [4]$$

This preprocessing has, on average, "empirically" led to the best results. The problem of choosing a *prior* preprocessing method is a very interesting and challenging one but was is of the scope of this paper.

4.3. PROCESSED DATA

The chart in *Figure 3* compares time series before and after preprocessing for Ciba-Geigy's stock prices. It shows that trends have disappeared and that local variations are now amplified. This is an advantage because it outlines the differences between increases and decreases and should improve the quality of the classification.

4.4. CLASSIFIER

We are facing a problem of pattern classification. The *Bayes decision theory* states that :

« to minimize the overall risk, [we just have to] compute the conditional risk

$$R(\alpha_i|x) = \sum_{j=1}^{k} \lambda(\alpha_i|\omega_j) P(\omega_j|x) \qquad \text{for } i = 1,\dots,a$$

and select the action α_i for which $R(\alpha_i|\mathbf{x})$ is minimum » [Duda and Hart, 1973]. [5]

This resulting minimum overall risk is called the *Bayes Risk* and is the best performance that can be achieved. k is the number of possible states of nature, a is the number of possible actions, $\lambda(\alpha_i|\omega_j)$ denotes the *loss* incurred for taking action α_i when the state of nature is ω_j, and $P(\omega_j|\mathbf{x})$, the *a posteriori* probability of ω_j, that is, the probability that the state of nature is ω_j conditioned on the observation of a particular pattern $\mathbf{x}$.

We also know from theory that in the case of a minimum-error-rate classification[2] where all errors are equally costly, the loss function becomes symmetric and the conditional risk given by the equation in rule [5] then reduces to :

$$R(\alpha_i|x) = 1 - P(\omega_i|x). \qquad [6]$$

Moreover, because our system performs a two-category classification, rule [5] and equation [6] can be simplified as follows :

« FOR MINIMUM ERROR RATE,
decide INCREASE if P(INCREASE|$\mathbf{x}$) > P(DECREASE|$\mathbf{x}$),
otherwise decide DECREASE. » [7]

This new rule points out the values needed to take position: the ***a posteriori* probabilities**.

[2] The case of minimum-error-rate classification, to which we restricted ourselves in this paper, can be easily extended to the general case.

Our goal is now to build a classifier able to provide estimates of these probabilities as output. Amongst the methods proposed to design classifiers, some of them requires knowledge of the form of underlying probability distributions, and are said to be parametric. The others do not make any assumption about the form of the underlying probability distributions, and are said to be non-parametric.

The common parametric forms rarely fit the densities actually encountered in practice. As we did not know the underlying distribution of our problem, we decided to use non-parametric techniques and investigated two of them:

- *Parzen Windows*, which estimates the *likelihood* density function $p(\mathbf{x}|q_i)$, from which we can derive *a posteriori* probabilities $P(q_i|\mathbf{x})$ [Duda and Hart, op.cit.] [Schalkoff, 1992].
- *Artificial Neural Networks* (ANNs) trained for directly estimating the *a posteriori* probabilities $P(q_i|\mathbf{x})$.

4.4.1. Parzen Windows

The Parzen Windows are a non-parametric technique that provides a non-biased estimate of the probability density function $p(\mathbf{x}|\hat{A}_n)$ that a particular sample **x** falls into a particular space region $\hat{A}_n$. In other words, it provides a non-biased estimate of the class-conditional probability density function $p(\mathbf{x}|q_i)$, where q_i is the class corresponding to the region $\hat{A}_n$.

Estimates are obtained by applying the following formula:

$$p_n\left(x|\Re_n\right) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{V_n}\varphi\left(\frac{x-x_i}{h_n}\right), \qquad [8]$$

where n is the number of samples falling into $\hat{A}_n$, V_n is the volume of region $\hat{A}_n$, $\mathbf{x}_i$ is a sample falling into $\hat{A}_n$ and φ is a window function centered at $\mathbf{x}_i$ and of size h_n. Estimates for $p(\mathbf{x}|\hat{A}_n)$ can then be interpreted as *interpolations of window function activations*, each sample contributing to the estimate in accordance with its distance from **x** [Duda and Hart, op.cit.].

Some conditions are required to assure that $p_n(\mathbf{x}|\hat{A}_n)$ is a "legitimate" density function (i.e. that it be non-negative and integrate to one) and converges to $p(\mathbf{x}|\hat{A}_n)$ in mean and variance as n approaches infinity. Four conditions on the window function are necessary to keep it well behaved. They are satisfied however by most density functions that we might think of using for window functions.

As we never dispose of infinity of samples in practice, we have to try to assure convergence as quickly as possible. Facing this problem, the choice of the volume V_n of the region $\hat{A}_n$ is crucial because of its major effect on $p_n(\mathbf{x})$. The Parzen Windows technique does not give a lot of indication about this choice: if V_n is too large, the estimate will suffer from too little resolution ; if V_n is too small, the estimate will suffer from too much statistical variability. With a limited number of samples, the best we can do is to seek some acceptable compromise [Duda and Hart, op.cit.].

The Parzen Windows technique gives us estimates of class-conditional probabilities $p(\mathbf{x}|q_i)$. To obtain *a posteriori* probabilities $P(q_i|\mathbf{x})$, we have to apply the *Bayes rule*, which is given by :

$$P\left(q_i|x\right) = \frac{p\left(x|q_i\right)P\left(q_i\right)}{p\left(x\right)}, \qquad [9]$$

where $P(q_i)$ is the *a priori* probability of class q_i and $p(\mathbf{x})$ is the scale factor which assures that *a posteriori* probabilities sum to one.

The *a priori* probabilities $P(q_i)$ are not known but can be estimated by dividing the number of samples falling in class q_i by the total number of samples. Doing that, we implicitly make the assumption that the number of samples is large enough to be representative. Since we are mostly interested in the comparison between these probabilities, the computation of $p(\mathbf{x})$ can be avoided.

The conditions on the window functions are, in particular, satisfied by the multivariate[3] Gaussian function that we use in our system:

$$\varphi_i\left(x\right) = e^{\left[-\left(x-x_i\right)^t\left(x-x_i\right)/2\sigma^2\right]}. \qquad [10]$$

The σ that appears here is the smoothing parameter of the window function[4]. It corresponds to h_n in equation [8] and influences V_n. We know that neither limiting case (σ or $V_n \rightarrow 0$ and σ or $V_n \rightarrow \infty$) provides optimal separation of the two classes. Moreover, there is no analytical method to compute optimal values for σ. Therefore, we used an averaging empirical method based on measures of the distance to the k^{th} nearest neighbor to compute *after the training* values for every pattern unit σ. This method allowed us to lower the classifier resolution in regions where the number of samples is low and to increase it where the number of samples is high (i.e. where it is needed).

Apart from σ, the only parameter to tune is the dimension of the input space, the order p mentioned above (see *2. From Prediction to Classification*). The lower the value of p, the lower the number of vectors needed for correct training. As p growths, we obtain more information on past prices evolution, but we also potentially get more noisy and less significant data. So, a compromise must be found between training set size and quantity of information. After several testing, a value of 10 has been elected.

4.4.2. Artificial Neural Networks

Artificial Neural Networks (ANNs) have been widely proposed as a potentially powerful approach to solve difficult problems such as vision, pattern and speech recognition [Hennebert *et al.*, 1994; Hertz *et al.*, 1991; Lippmann, 1987]. In the framework of pattern recognition, ANNs are a non-parametric approach because there is no need for any particular assumptions about statistical distributions and independence of input features, i.e. there is no a priori relation between an ANN's parameters and a presupposed statistical distribution. This aspect is the major strength of ANNs used in classification.

Amongst the plethora of different ANNs structures that have been suggested, the Multi-Layer Perceptron (MLP) has probably been the best studied class of neural

[3] Parzen's results which are limited to the univariate case have been extended by Cacoullos to cover the multivariate case in the special situation that the multivariate kernel (category) is a product of univariate kernels [Cacoullos, 1966]. Equation [10] satisfies this additional condition.

[4] This window function is also used by Specht, who proposes an implementation of the Parzen Windows technique in a form resembling to a neural network and called Probabilistic Neural Network [Specht, 1988 and 1990].

network. It has been shown in recent research that this kind of network can be trained to produce any desired outputs for given inputs [Barron, 1993]. MLPs can then be seen as universal function approximates.

In [Bourlard, 1990], it has been proved that, under conditions, it is possible to train a MLP to generate *a posteriori* probabilities $P(q_i|\mathbf{x}_n)$ when a particular pattern $\mathbf{x}_n$ is provided to its input. The training data set consists of a labeled sequence of N patterns $\{\mathbf{x}_1, \mathbf{x}_2, \ldots , \mathbf{x}_N\}$ and the training of the parameters is based on the minimization of the following Mean Square criterion,

$$E = \frac{1}{2} \sum_{n=1}^{N} \sum_{i=1}^{k} \left[g_i(x_n) - d_i(x_n) \right]^2 \qquad [11]$$

where $g_i(\mathbf{x}_n)$ represents the i^{th} output value given $\mathbf{x}_n$ at the input, and $d_i(\mathbf{x}_n)$ is the associated target value, which is equal to δ_{ij} (Kronecker Delta) if the input is known to belong to class q_j ("*1-from-k*" training).

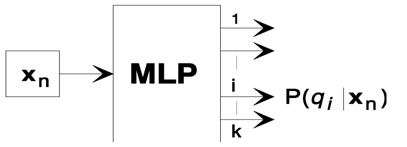


Figure 5 - MLP used as a posteriori probability estimator.

To obtain estimates of the *a posteriori* probabilities, three conditions have to be satisfied:

1. Use a particular training method which associates each output unit with a particular class q_i of the set of classes Q = $\{q_1, q_2, \ldots , q_k\}$;
2. Assure that the MLP contains enough parameters (i.e. hidden units) ; and
3. Reach the global minimum of E (in equation [11]).

MLPs have layered feedforward architecture. They have an input layer, zero or more hidden layers and an output layer. Each layer is composed of one or more units. Units of the first layer simply distribute input values to each unit of the first hidden layer, and units of other layers perform a non-linear function of its inputs, i.e. the so-called transfer function, that is typically a sigmoid:

$$f(x) = \frac{1}{1 + e^{-x}} . \qquad [12]$$

For our system, the number of units in the input layer is equal to the order p and the number of output units is equal to the number of classes (2). Each layer (except the input layer) is connected to the previous via a weight matrix according to :

$$y_i^L = f\left(\sum_j w_{ij}^{L,L-1} y_j^{L-1} \right), \qquad [13]$$

where y_i^L is the output of unit i in layer L and $w_{ij}^{L,L-1}$ is an element of the weight matrix between layers *L-1* and *L*.

For the reason explained in *4.4.1.*, a 10-unit input layer has been elected. We used a network with one hidden layer of 12 units. One layer is sufficient to approximate complicated functions and 12 units is the configuration that we chose after several testing because it has given reasonable results. The classical back-propagation algorithm has been used to minimize error during training.

4.5. Investment Strategy

When planning to invest in the stock market, computing the “mean square error” or analyzing graphs of inputs are not the most important elements. We believe that they are only tools, and not goals.

What does interest us is the ability to make real investments and to see if the system can beat the market. Therefore, we built an investment strategy based on the output probabilities to decide whether we should buy or sell the stock. Final results are given in Swiss Francs and are to be compared with the performance of the buy and hold strategy (buy on first day and hold stock during the hole period).

On the basis of the probabilities, we compute a ratio, denoted δ, given by:

$$\delta = \frac{P(increase|x) - P(decrease|x)}{P(increase|x) + P(decrease|x)}, \qquad [14]$$

where **x** is the input vector. If we do not hold the stock and if delta is higher or equal to a defined value, called “buying epsilon”, we buy the stock. If we already hold the stock and if delta is lower or equal to a defined value, called “selling epsilon”, we sell the stock. Otherwise, we do not change our position. The strategy is summarized in the following table.

POSITION	DELTA		
	^{3} BUYING EPSILON	£ SELLING EPSILON	OTHERWISE
NO	BUY	--	STAY
YES	--	SELL	STAY

Figure 6 - Summary of the investment strategy.

The “epsilon” parameters allow us to take into consideration the degree of confidence required by the investor. In particular, its level of risk aversion can influence it: the higher its level, the higher the buying and selling epsilon.

Our strategy is a short-term one: it can decide to buy one day, sell the next, and re-buy the day after. So, it can potentially generate a lot of transaction costs. We did take them into account, using an average cost percentage of 0.5% on each transaction.

During periods where we do not hold any stock, we place the money at the bank in a current account, which yields an arbitrarily chosen return of 4% per year.

5. Results

We will present the results in three steps, going from theory to practice. The first step gives an idea of the classification quality. The second one shows the influence of tuning investment strategy parameters. And finally, the third one presents cash results obtained when applying the investment strategy.

5.1. Hit Rates

The following table gives a summary of the “hit rates” obtained with both classifiers.

	Nestlé	**Ciba-Geigy**	**Sandoz**	**Zurich**	**UBS**
Parzen Window	63.06	62.87	61.30	59.14	61.89
MLP	61.30	54.03	50.49	47.15	48.13

Figure 7 - Comparison of hit rates between the Parzen Window classifier and the MLP classifier (in %).

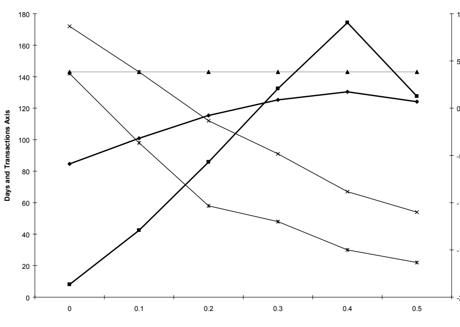
As far as the comparison is concerned, there is no doubt: the Parzen Window classifier gives far better results than the MLP classifier. The average difference is about 10%.

The Parzen Window classifier obtains high hit rates. In average, they are above 61%, which is one of the highest rates we have encountered in the literature. Moreover, they are quite constant (between 59 and 63%). This means that the Parzen Window classifier manages to anticipate future fluctuations of the processed time series with a considerable rate of success and a relatively high degree of confidence.

On the contrary, the MLP classifier does not give very satisfactory results. Hit rates are in average around 52%, which is not far from pure hazard, and their range of fluctuation is large (nearly 15%). However, we believe that spending more time to adjust structure and learning parameters of the MLP and/or using a less fragile method than back propagation (e.g. quasi-Newton) to minimize error could sensibly improve the results.

5.2. INVESTMENT STRATEGY PARAMETERS

The δ ratio computed in the investment strategy is compared to two parameters: the buying epsilon and the selling epsilon. The following figures show the influence of different values for these parameters[5].



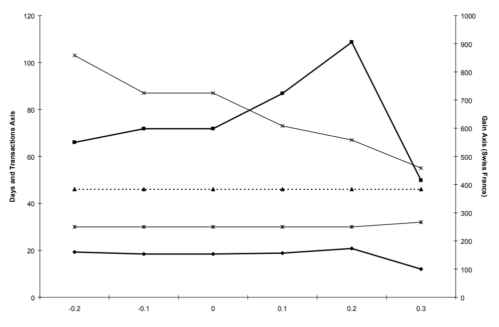


Figure 8 - Influence of the buying epsilon parameter. Figure 9 - Influence of the selling epsilon parameter.

Some explanations are useful to understand the legend. The "Number of Playing Days" is the number of days during which we hold the stock. The "Number of Transactions" is the number of times that we buy or sell the stock. The "Classifier Net Gain" is the gain (or the loss) realized when using the system *minus* the costs generated by transactions and *plus* the interest of the money placed on a bank account during non-playing periods. The "Classifier Potential Gain" is the gain (or the loss) that would potentially be obtained if we could play on the whole test period (509 days) in the same way (i.e. as well or as badly) that we do on the actual playing days. The "Buy and Hold Gain" is the gain (or the loss) realized when applying the "Buy and Hold Strategy" (buy on the first day, sell on the last).

[5] We give here the example for the Parzen Window classifier and the Nestlé's stock.

It appears as a horizontal straight line on the figures because it is not influenced by the epsilon parameters.

The main lesson we can learn from *Figure 8* is that the buying epsilon parameter influences greatly the classifier gains, and that it exists an optimal value that maximizes gains (0.4 on the figure). So, it appears that to a certain point our interest is to invest only when the degree of confidence of the classification is large enough. We can also see that the number of playing days and the number of transactions decrease as the buying epsilon becomes larger. This is logical: the higher the buying epsilon, the more careful we are; and the more careful we are, the less we buy the stock.

The same conclusion can be drawn from *Figure 9*: it exists an optimal value of the selling epsilon parameter that maximizes the classifier gains (0.2 on the figure). However, if the effect on the number of playing days and the classifier potential gain is still important, the effect on the two other values is less significant. Thus, it appears that tuning the selling epsilon parameter allows to maintain the classifier net gain while decreasing the playing period.

We also encountered cases where choosing a negative selling epsilon can improve performances. The reason of this additional gain is here not a decrease of the playing period but an increase of the classifier net gain due to lower transaction costs (i.e. the system sells the stock less often).

5.3. Cash Results

The following table presents a summary of the results. Interesting comparisons are between "classifier" gains and "buy and hold" gain and between "Parzen Window" gain and "MLP" gain.

Stock	Classifier	Buying Epsilon	Selling Epsilon	Number of Playing Days	Number of Transactions	Classifier Net Gain	Classifier Potential Gain	Buy and Hold Gain
Nestlé	Parzen Window	0.8	0.0	17	6	107.78	1566.28	194.02
	MLP	0.1	0.0	42	12	94.90	511.93	
Ciba-Geigy	Parzen Window	0.5	-0.1	83	24	72.47	234.42	135.67
	MLP	0.1	0.0	7	4	122.71	5988.75	
Sandoz	Parzen Window	0.3	0.1	201	60	25.47	9.79	129.6
	MLP	0.4	0.0	452	12	122.84	133.83	
Zurich	Parzen Window	0.4	0.2	67	30	173.09	905.58	383.88
	MLP	0.0	0.0	509	2	383.88	383.88	
UBS	Parzen Window	0.4	-0.3	116	22	21.55	-81.17	488.89
	MLP	0.3	-0.1	304	2	263.77	406.66	

Figure 10 - Comparison of both "classifier" cash performances and the "buy and hold" performance.

A first observation is that the Parzen Window classifier gives better results than the MLP classifier on 3 stocks (out of 5), but that the difference is less than for hit rates. High hit rates do not seem to automatically lead to high gains. This means that a good decision on the processed data - on which the hit rate is computed - may not lead to a good decision on the raw data - on which the gains are computed.

The responsible for this is the data preprocessing. Removing the trend from time series makes them easier to handle by a classification system. But, it also takes away some valuable information and causes the system to buy stocks when identifying a positive fluctuation even if the trend is highly negative (and conversely). We will see further how this problem can be solved.

We can also observe on the table that if classifier net gains are always lower than buy and hold gains - but this is not a sign of bad performance since the first are realized in less days than the second -, classifier potential gains are higher than buy and hold gains in 60% of the cases. This means that if we manage to repeat on the entire test period the same rate of performance as on the actual playing period we can potentially make substantial profits.

However, we have to stay careful when analyzing potential gains. To extend the playing period performance to the entire test period, we will have to find a portfolio of stocks with more or less matching playing periods and to build a portfolio management system to invest each time in the "predicted more interesting" stock. Since the number of needed stocks can be very high (especially if we consider that high potential gains are very often realized on short playing periods) and overlapping playing periods will be difficult to avoid (and will possibly reduce individual performances), it promises to be difficult to set up such a system.

One last important observation is that optimal epsilon values shown on *Figure 10* seem to depend mainly on the stock and that prior optimal values seem difficult to find. We are convinced however that improving the link between high hit rates and high gains will allow us to tune analytically epsilon values according to hit rates (in addition to the risk aversion level of the investor): the higher the hit rate, the closer to zero the buying and selling epsilon will be.

6. CONCLUSIONS

The system presented in this paper had two main goals. First, it had to provide valuable information on future stock prices movements. Second, it had to be a good pattern recognition system. At least the second of these goals is reached. The system has a high rate of success (more than 61% for the Parzen Window classifier) on identifying a future increase or decrease of the processed time series, that is, on identifying fluctuations around the trend.

Although we still believe that a posteriori probabilities of price movements is a valuable information, we did not satisfactorily reached the first goal because this information alone is not sufficient and information about the trend appears to be necessary as well. We did obtain however some encouraging results that confirm us in the idea that the probabilistic approach is a good one. With some improvements and further developments, it may well lead to an efficient investment tool.

In its current form, our system can already be considered as a decision support system that gives investment advice. It has two main advantages: the first is that it gives investors a qualified information with probabilities and the second is that it takes into consideration the risk aversion level of investors.

7. FUTURE WORKS

7.1. NUMBER OF CATEGORIES

In this paper, we limited ourselves to a two-category classification. We saw that it could be interesting to obtain more information on the increase or decrease scale to avoid position changes when movements are not significant. Using three categories (increase, decrease and "stability") instead of two, or four (strong increase, weak increase, weak decrease, strong decrease) could provide us with more precise information about the future stock price movement. By replacing prediction with classification, we have gone from one extreme to the other. An optimal position is certainly to be found between both extremes, and progressively adding categories to our system is certainly a way to find it. Another potential improvement could be to attempt the modeling of transitions between categories, introducing double stochastic models such as Hidden Markov Models [Rabiner, 1989] or other approaches recently proposed in speech processing [Fontaine et al., 1994].

7.2. PREPROCESSING

A major way to improve on the system is to draw more information of the preprocessing. The idea is to use the moving averages computed to process raw data and to train a second classifier (identical to the first) with them. Considering that the moving average series is a smoothed version of the raw data series and contains information about short-term trends, the second classifier should provide us with *a posteriori* probabilities of trend movements (see *Figure 11*).

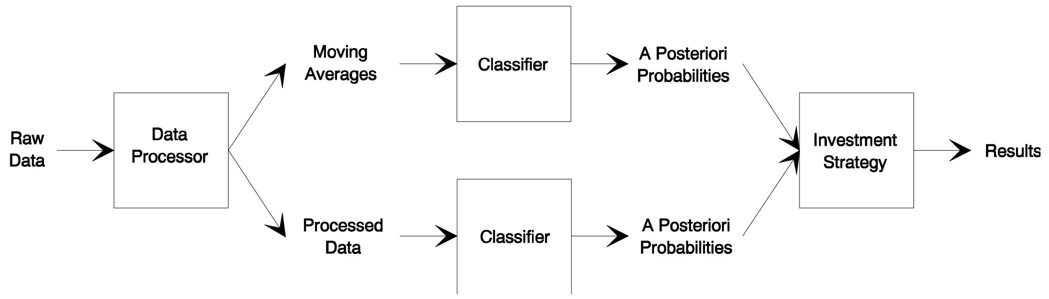


Figure 11 - Elements of the classification system after improvement of the preprocessing.

Combining outputs of both classifiers in an extended investment strategy should then allow the system to take more sensible investment decisions and prevent from buying stocks when identifying a positive fluctuation around a highly negative trend (and conversely). The main advantage of this new system should be to improve the translation of high hit rates into high gains.

7.3. INPUT DATA

Non-satisfactory results mainly come from one (or both) of the following two causes: a bad-quality system and bad-quality input data. This paper has dealt so far with the first cause and we will now briefly focus on the second.

Two kind of information are used on financial markets. On one hand, there is information coming from *fundamental analysis*. It consists of indicators such as the risk-free interest rate or future earnings and dividends, which are known to have an influence on the stock market.

On the other hand, there is information coming from *technical analysis*[6], which is based on past stock price and transaction volume analysis. Although a large part of the methodology of technical analysis lacks a strictly logical explanation [Rosenfeld, 1975], recent researches have shown that some of the most popular chartist techniques - which are the main part of technical analysis - work quite well (see *1. Introduction*).

The two approaches are often differentiated in the following way: "fundamentalist" tends to look forward and "technician" backward [Sharpe and Alexander, 1990]. Thus, using past stock prices as input to our system, we *implicitly* performed a technical analysis.

Useful additional input data can be obtained in two ways. The first one consists of adding some of the main fundamental analysis indicators. This should allow the system to take into consideration some factors external to the market but which are reflected in it.

The second way consists of using *explicitly* some technical analysis results. Here again we have two possibilities. We can replace past price with technical indicators[7] or we can use technical analysis to generate artificially additional training vectors that will be learned by the classifier. This second technique is called "learning from hints" and can significantly improve results in cases where some information or expertise about the problem is already available [Abu-Mostafa, 1990 and 1994].

REFERENCES

R. Duda, P. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons, Inc., 1973.

R. Brealey, S. Myers, *Principles of Corporate Finance*, Fourth International Edition, McGraw-Hill, 1991.

The Economist, *A Survey of the Frontiers of Finance, The Mathematics of Markets*, October 9th, 1993, pp. 5-20.

A. Kouam, F. Badran, S. Thiria, *Approche Méthodologique pour l'Etude de la Prévision à l'aide de Réseaux de Neurones*, Conférences Neuro-Nîmes 91, 1991, pp. 117-127.

H. Bourlard and C. Wellekens, *Links between Markov Models and Multi-Layer Perceptrons*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 12, IEEE, 1990, pp. 1167-1178.

Lippmann, *An Introduction to Computing with Neural Nets*, IEEE ASSP Magazine, 1987.

J. Hennebert, M. Hasler and H. Dedieu. *Neural Networks in Speech Recognition*, 6th Microcomputer School, invited paper, Prague, Czech Republic. 1994. pp. 23-40.

[6] For more details on technical analysis, see [Shaw, 1975] or [Cohen et al., 1987].

[7] Replace instead of add for two reasons. First, technical indicators are based on past price analysis and therefore contain the same information as past prices but increased by a first processing. Second, adding technical indicators would raise the input space dimension to a level where it becomes almost impossible to find enough different training vectors to efficiently fill the input space.

Hertz *et al.*, *Introduction to the Theory of Neural Computation*, Lecture Notes Volume in the Santa Fe Institute Studies in the Science of Complexity, Addison-Wesley Publishing Company, 1991.

D. Specht, *Probabilistic Neural Networks*, Neural Networks, Vol. 3, Pergamon Press, 1990, pp. 109-118.

D. Specht, *Probabilistic Neural Networks for classification, mapping, or associative memory*, Proceedings IEEE International Conference on Neural Networks, 1, 1988, pp. 525-532.

A. Refenes, M. Azema-Barac, P. Treleaven, *Financial Modeling using Neural Networks*, UCL-CS, RN-92-94, University College London, Department of Computer Science, 1992.

R. Schalkoff, *Pattern Recognition: Statistical, Structural and Neural Approaches*, John Wiley & Sons, Inc., 1992.

T. Cacoullos, *Estimation of a multivariate density*, Annals of the Institute of Statistical Mathematics (Tokyo), 18(2), 1966, pp. 179-189.

Takens, *Detecting Strange Attractors in Turbulence*, Lecture Notes in Mathematics, Springer Verlag, New-York, 1980, pp. 366-381.

C. Ris, H. Dedieu, M. Hasler, *Order Estimation and Nonlinear Prediction with Radial Basis Functions*, Proceedings Eusipco 94, Edinburgh, 1994, pp. 1492-1495.

A. Barron, *Universal Approximation Bounds for Superpositions of a Sigmoidal Function*, IEEE Transactions on Information Theory, Vol. 39, No. 3, May 1993.

A. Shaw, *Technical Analysis*, in S. Levine, ed., *Financial Analyst's Handbook I*, Homewood Ill.: Dow Jones-Irwin, 1975, pp. 944-988.

J. Cohen, E. Zinbarg, A. Zeikel, *Investment Analysis and Portfolio Management*, Fifth Edition, Homewood Ill.: R. Irwin, 1987, Chapter 8.

F. Rosenfeld, ed., *The Evaluation of Ordinary Shares*, a summary of the proceedings of the Eighth Congress of the European Federation of Financial Analysts Societies, Dunod, Paris, 1975, p. 297-298.

W. Sharpe, G. Alexander, *Investments*, Fourth Edition, Prentice Hall International Editions, London, 1990, pp. 683-690 and 696-701.

Y. Abu-Mostafa, *Learning from Hints in Neural Networks*, Journal of Complexity 6, Academic Press Inc., 1990, pp. 192-198.

Y. Abu-Mostafa, *Financial Market Application of Learning from Hints*, in A. Refenes, ed., *Neural Networks in the Capital Markets*, Wiley, England, 1994.

L. Rabiner, *Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*, 1989, Proceedings of the IEEE, Vol. 77, No 2, pp. 257-285.

V. Fontaine, J. Hennebert and H. Leich, *Influence of Vector Quantization on Isolated Word Recognition*, Proceedings Eusipco 94, Edinburgh, pp. 115-118