Prediction, Volatility and Complexity of Financial Markets: a Move-Based Approach

Anne-Sophie Bratel

MSc by Research in Pattern Analysis and Neural Networks

ASTON UNIVERSITY

September 2008

Acknowledgment

I express my gratitude to **Pr. Ian.T. Nabney**, for his help, his guidance, his patience and his kindness during the course of the project. I would like to thank Christophe Agopian for giving me advices and providing me with the data. I am also grateful to my fellow MSc. students and the PhD students of the NCRG for their assistance, their support and for the good time we spent together.

ASTON UNIVERSITY

Prediction, Volatility and Complexity of Financial Markets: a Move-Based Approach

Anne-Sophie Bratel

MSc by Research in Pattern Analysis and Neural Networks, 2008

Thesis Summary

We consider financial time series by representing the movements in the data rather than the prices at fixed time steps. More precisely, we first identify a set of discrete levels and then record the first time the process crosses a level. This information can be simplified still further by simply recording the direction of the movements, creating a binary string: 1 for "up" and 0 for "down". We show that it is possible to use this representation in order to create effective forecasting models based on the distance between the different sequences. Their parameters are determined by cross validation, the maximum likelihood and the data complexity. Afterwards, we construct several trading strategies based on the predictions given by the model, then, we measure the trading rules performances and compare them to a Buy and Hold strategy.

Keywords: Prediction, binary strings, Volatility, Complexity, Financial Markets

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Efficient Market Hypothesis developed in 1965 by Fama [11] asserts that the current price of an asset already contains all information from the past prices; however, forecasting future values of financial series is of obvious interest in empirical finance, and there is plenty of practical evidence that forecasting can be successful. Numerous models involving sophisticated techniques such as neural networks, genetic algorithms, ARMA and GARCH models, already exist to reproduce the behaviour of price evolution with greater or lesser accuracy. While the complexity of the forecasting methods increases, financial organisations are seldom able to generate a very accurate prediction of asset prices.

Therefore, in this project, we evaluate a new approach: we study financial markets by representing the movements in the data rather than the prices at fixed time steps. Thanks to this process, we can describe the behaviour of the price in a different way: we define a set of levels instead of a set of times and record the times at which they are reached. Then we condense the recent past into a sequence of binary digits where "1" means up and "0" means down and we try to estimate the probability for the next bit to be an up. We then build forecasting models for this set of strings; we consider them as patterns and we compute the expected next bit by comparing distances with the other sequences; the predictions given by the model are then transformed in trading strategies.

As our purpose is to seek an efficient financial forecasting model using patterns in

the data, we start by introducing the efficient market hypothesis issues and presenting some existing models. Then we will present and analyse the data which are used for all the experiments: 10 years of daily records of main stocks indexes. Using cross validation, maximum likelihood and maximum entropy, we manage to determine the parameters of our models. Finally, their outputs permit us to build trading rules and we compare their performances with the Buy and Hold strategy.

1.1 Plan of the Thesis

Chapter 1 is this introduction.

Chapter 2 is an overview of financial forecasting and explains the processes we will apply to the data in order to transform them into binary strings.

Chapter 3 describes the stock index series and explores their statistical properties.

Chapter 4 looks at the statistical properties of the string time series and determines the optimal length of the words using the principle of maximum entropy.

Chapter 5 investigates different methods to estimate the parameters of the model such as cross validation and the maximum likelihood and describes the classifications obtained using neural networks.

Chapter 6 discusses prediction accuracy and trading rules measurements.

Chapter 7 concludes the thesis with a discussion of the major results and some suggestions for future work.

Chapter 2

Presentation of the model

2.1 The efficient market hypothesis

The main purpose of this project is to predict the future behaviour of stock index time series, but it takes place in an old financial controversy: can we use past data to predict an asset's future prices?

Intuitively, it seems logical that price history influences the current price of a stock, nevertheless, the efficient market hypothesis (first stated 30 years ago) asserts that the present value of an asset encompasses all the available information. It means that future prices are independent of past information. Louis Bachelier (1870-1940) was the precursor of this theory and found out that the price returns can be modelled by Brownian motion under the following assumptions [30]:

- Let $P(y)$:price of the stock at time y,
- Let $P(t+y)$: price of the stock at time $t+y$,
- If $\frac{P(t+y)}{P(y)}$ is independent for all prices up to time y,
- If $\log(\frac{P(t+y)}{P(y)})$ follows a normal distribution with mean $\mu * t$ and variance $\sqrt{t} * \sigma$,

then P follows a geometric Brownian motion with mean μ and volatility σ. This phenomenon is a continuous time stochastic process with continuous state space or a

Markov model which assumes that future states of the machine depend only on the current state and have no link with earlier states: the increments are stationary, independent and Gaussian but Brownian motion is not differentiable anywhere. Although this model may be convenient to describe the movement of market indices in the short run but it is useless in the long run.

Thus, if we assume that a security's price follows geometric Brownian motion, it means that only the present value affects the evolution of future prices. For instance, this also implies that the probability that a given stock doubles its price in the following month is the same if its present value is 5 or 55: this hypothesis seems not to reflect a realistic market. The efficient market hypothesis has numerous variants as the so-called martingale hypothesis: the present price of a security is the fair price in the sense that the expectation of the present value of a future price is equal to the present price, which implies that the best estimate of future value is the current value. Moreover, there are some specific cases for which it is proven that price sequences do not follow a geometric Brownian motion. Indeed, if we consider crude oil time series, a new model can be built under "the assumption that the future resembles the past and a risk neutral valuation based on the new model" [29]. We can therefore remark that security price data are not all consistent with the efficient market hypothesis. In particular, many traders believe that the prices of some securities tend to revert to fixed values: when the current price is less than a threshold, the price is more likely to increase and when it is greater it tends to decrease. This phenomenon is called mean reversion and it can be explained by an AutoRegressive Model (AR).

Benoit Mandelbrot (known for his studies in fractals) demonstrated in 1963 that the assumptions of Wiener Brownian motion cannot be applied to a real stock market because of the 'Noah' and 'Joseph' effects [7]. He found out that price sequences are discontinuous and have an infinite variance: every time a price suffers a strong discontinuity, a new point appears on the distribution tail and these big movements are concentrated in time. Furthermore, he proved that the return distributions are not Gaussian, but are sharp-peaked and heavy tailed but L-Stable [7]. He called this

phenomena the Noah effect. Later he demonstrated the cyclic aperiodic behaviour of economic evolution: in 1975, he published a new model taking in account the role of long run memory in financial time series. He proved that returns follow a cyclic but non-periodic behaviour that he called the Joseph effect; afterwards, he mixed these two models to produce a new one called multi-fractional Brownian Motion.

To conclude, it was shown that there are useful patterns in past data; therefore, many people think it could be reasonable that a stock's recent price history can be used in financial forecasting. Consequently, a new line of research has emerged in data technical analysis since the 90s and these new methods permit building simple trading rules which are now used in the stock exchange every day.

2.2 Forecasting methods

During the past few years, a lot of financial prediction models have been built but only a few of them achieved good predictions which can be applied in the real stock market. Most of these models use new techniques such as time series models, neural networks, machine learning or genetic algorithms, and consequently most technical analysis are mainly based on past price observations.

An example of the application of simple trading methods such as Moving Average and Trading Range Break could be found in the studies by Brock, Lakonishok and LeBaron in 1992 [34]. These indicators lead to very easy and popular trading strategies which permit making direct profits in the stock exchange. Many other models have been developed with more or less success to estimate price evolution: new trading rules appeared as the "relative strength" [21] or the "buy and hold" which consists to purchase an asset at the beginning of a period and to keep it until the end. Some other forecasting methods have been developed using chartist analysis: according to Murphy(1999) and Béchu & Bertrand(1999) [14], it is possible to recognize some non-linear configurations which can be used in trading strategies.

Price prediction algorithms are also used for trading: in 1997 Mitchell [31] used common learning algorithms while Muller [16] tried a new and powerful algorithm, the

support vector machine. During the same period, Kutsurelis and Cheung [13] used neural networks with back-propagation to predict equity price evolution. Moreover genetic algorithms are also applied to financial markets and often mixed with neural networks in order to achieve good short term predictions. One of the features of these processes is the fact that they are constantly learning and so improving, they can adapt themselves to all the current situations.

Although all these methods are very sophisticated, they can not really achieve good predictions, they can not guess the exact future value of a stock and their forecasting capacity is very limited. Nevertheless, they are accurate enough to build more complex trading rules which make profits; that is one of the reasons why we decided to look at another aspect of the problem in this project. Instead of forecasting stock future prices, we will try to predict the next movement: basically, if the index value is increasing or decreasing in the next days or weeks depending on the prediction's accuracy.

2.3 Binary string models

In this section we will describe the new forecasting method introduced by Dupire from which we based our study [6].

2.3.1 A new approach

As we explained in the problem statement, the classical way to summarize the information contained in a financial time series is to define a set of times at which to record the asset price. Then we apply algorithms to this data in order to predict the future values; Dupire's approach was different, in that he defined a set of levels and recorded the times at which these levels are reached. Indeed, he studied financial markets by representing the movements in the data rather than the price at fixed time points.

2.3.2 Cross points and binary strings

Dupire defined a set of levels which are equally spaced and then recorded the times at which a level different from the last one is reached. More precisely, let us suppose we are on a time interval $[0;T]$ and n levels. We first record the time when the curve crossed first a level l_i, then we save the date t_1 and the associated price at t_1, l_i. Afterwards, we are looking for the next reached floor either immediately above l_{i+1}, the same l_i or below l_{i-1}. We save again the data (t_2, P_2) with $P_2 = l_{i+1}$ or $P_2 = l_{i-1}$, depending on the path behaviour, and then carry on this process. At the end, we obtain a discretization of the information in the original data series. By linking all these recorded points, we can construct a staircase function E.

Considering this representation, we are able to record the information in a different way: if we use the convention $up = 1$ and $down = 0$, we can represent the curve's movement as a binary string. When the function E is going up we record a 1 and when E is going down we code a 0; this succession of up and down movements contains all the information we need.

The following figure gives a good illustration of this process. It shows the original price time series, the levels (10 in this case), and the function E constructed as we described. The binary string derived from E is given in the title. It is observed that the curve reach first the level 4400. Therefore this point is recorded. Then the path again crosses the level 4400, the data is also saved but we do not take it into account in the binary string because no new level is reached. Afterwards, a new floor is attained, the level 4700. We record the point and we write the first 1 of the binary string. As the next level reached is 5050, we write a second 1. But then, the next floor attained is 4700. Here the function E is decreasing so we code the first 0. We pursue this process to build all the binary sequence of this example.

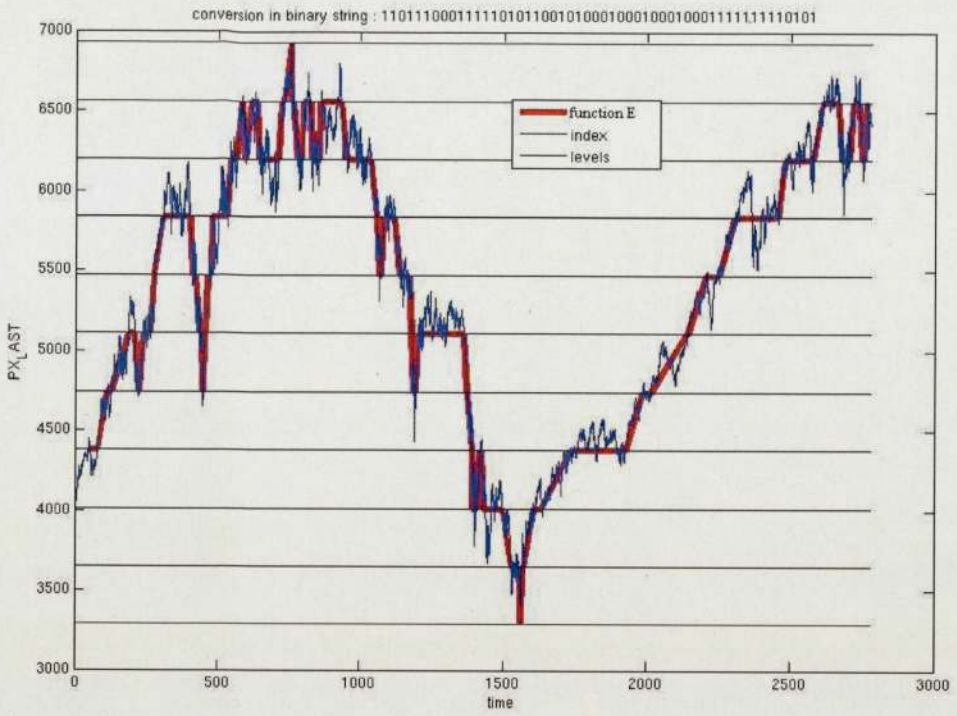

Figure 2.1: Example of binary string's construction.

2.3.3 Forecasting by non-parametric estimation

Once we have obtained a binary string, we choose a window length and we construct a series of delay vectors. We observe that these vectors are similar to DNA sequences coding a protein: each sequence means a different protein and thus it is possible to recognize the string which produces a particular protein. In our case, we would like to select patterns which predict an "up" using non-parametric estimation. It is therefore essentially a weighted kernel regression model, in that it consists of a set of past strings and their outcomes. For a new point we apply Dupire's theory by computing distances to those past strings and then weight their outcomes accordingly.

We have to define a distance between the sequences, because we require the predictions of nearby vectors to be substantially weighted and distant vectors almost ignored. We thus need a criterion that expresses the degree of similarity or difference of two strings. Two strings differing by one bit will be considered similar if the differing bit

is in further back, but if the differing bit is in the position corresponding to the most recent price change, the two strings will be judged as more different. Let us suppose we have the three following strings, A is the current string:

$$--------\text{>time}$$

A=011101,

B=111101,

C=011100.

B is closer to A than C because more recent bits match.

Dupire adopted a weighting scheme with exponential decay and defined the distance between two strings X and Y by the following formula:

$$d(X,Y) = \sum_{j=1}^{N} \alpha^{N-j} * (X_j - Y_j)^2, 0 < \alpha < 1. \tag{2.1}$$

We still have to estimate the influence of strings similar to the current one on the prediction of the next binary bit.

Dupire proposed the following model where each past string votes for its next bit with its own future weighted by its distance with the current strings.

$$p = \frac{\sum e^{-\lambda d(X,Y_i)} * p_i}{\sum e^{-\lambda d(X,Y_i)}}, \tag{2.2}$$

where $p_i = 0$ or 1 is the digit following Y_i and p is the computed probability for the current string to be followed by an up.

2.3.4 Another way to use binary strings

Lutz Molgedey and Werner Ebeldling [35] [17] [23] also use binary strings for local prediction of financial time series but unlike in Dupire's work they keep the notion of time. They consider small pieces of the returns record and build digit sequences by considering the values of returns over a few days.

Returns are defined by:

$$R_t = \frac{P_t}{P_{t-1}} \tag{2.3}$$

The alphabet is composed of three digits :"0","1","2" depending of the return at time (t). Afterwards, using conditional entropy, they achieve convincing prediction of the next move. While their way to construct the bit sequences is totally different, their process is quite interesting, especially about the concept of uncertainty, that is how last move predicts the next one using the minimisation of the conditional entropy. Besides, they applied this model to other fields such as human writing, images and astrophysics.

2.3.5 Conclusion

To conclude, Dupire can create effective forecasting models that achieve good profits with simple trading rules. One of these strategies consists in buying assets if $p > 0.6$ and selling them if $p < 0.4$.

Chapter 3

Data Analysis

The aim of this chapter is to present the data used in this project and explore its basic statistical properties.

3.1 Data presentation

To carry out empirical experiments, 5 sets of daily closing data for the main equity indices were used. It was decided to study the evolution of indices rather than stock price records because they are less dependent on a specific economic environment. Moreover, using this type of data avoids many distracting technical problems like no-quotation days or dividend payments. This data are from the Dow Jones EUROSTOXX 50 Index, FTSE 100 Index, Standard and Poor's 500 Index, Russell 2000 Index and NASDAQ-100 Index. In order to simplify the code and the figures in Matlab, the data are considered as a function of working days. The financial time series are defined as following:

- **The Dow Jones EUROSTOXX 50 Index (SX5E Index):** this is the leading blue-chip index for the Euro area, covering 50 stocks from different European countries.

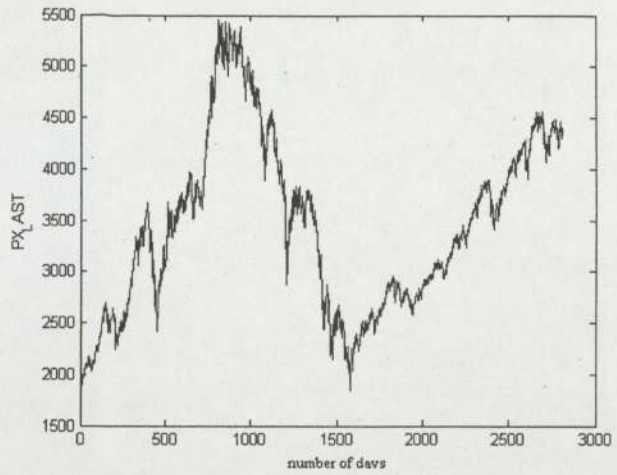

Figure 3.1: The Dow Jones EURO STOXX 50 Index between 1997 and 2007.

- **The FTSE 100 Index (UKX index):** The Financial Times Stock Exchange (FTSE) is an index of the 100 most highly capitalised companies listed on the London Stock Exchange. It represents about 80% of the capital of the whole London Stock Exchange and is by far the most widely used UK stock market indicator.

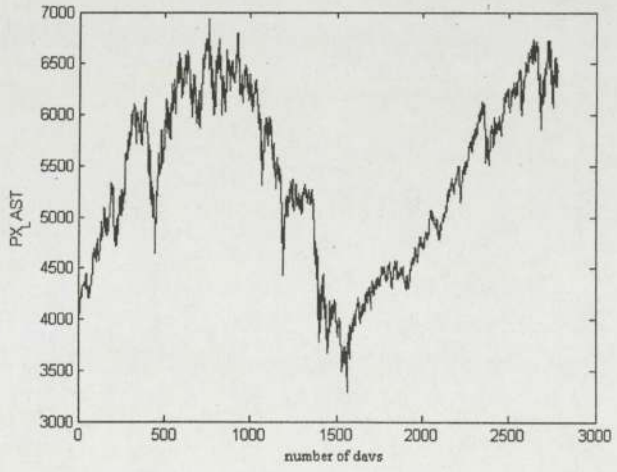

Figure 3.2: The FTSE 100 Index between 1997 and 2007.

- **The Standard and Poor's 500 Index (SPX Index):** The S&P 500 is an index containing the stocks of 500 Large-Cap corporations, which represents 70%

of all US publicly traded companies. All of the stocks in this index are those of large publicly held companies and trade on the two largest US stock markets, the New York Stock Exchange and the Nasdaq. After the Dow Jones Industrial Average, the S&P 500 is the most widely watched index of Large-Cap US stocks and it is usually considered as the benchmark for U.S. equity performance.

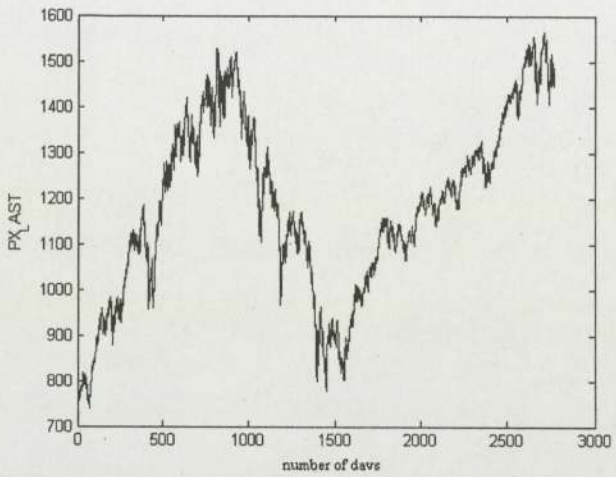

Figure 3.3: The Standard and Poor's 500 Index between 1997 and 2007.

- **The Russell 2000 Index (RTY Index):** The Russell 2000 is used to measure the performance of U.S. small companies stocks ("small caps"). It focuses on smaller capitalization companies and is quite diversified. This is the most quoted index that focuses on this portion of the economy, this index is quite volatile both in composition and in valuation.

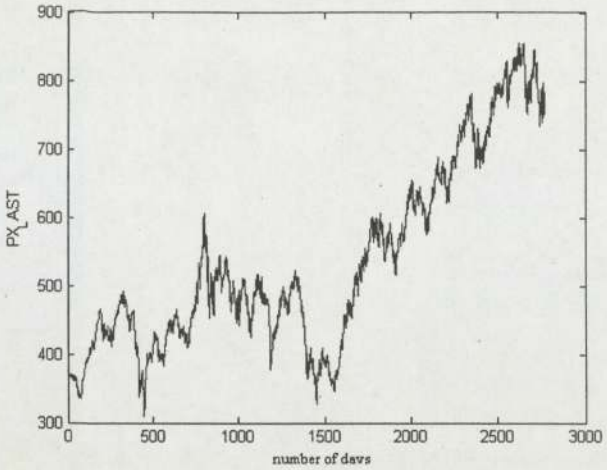

Figure 3.4: The Russell 2000 Index between 1997 and 2007.

- **The NASDAQ-100 Index (NDX Index):** The National Association of Securities Dealers Automated Quotation (Nasdaq 100) tracks the 100 largest stocks listed by the Nasdaq exchange. Nasdaq's companies tend to be smaller and younger than New York Stock Exchange companies, but the exchange also lists Dow Jones Industrial Average giants like Microsoft or Intel. This index is often treated as a "tech stock" index simply because its components are mostly new technology companies. Moreover this index can be extremely volatile.

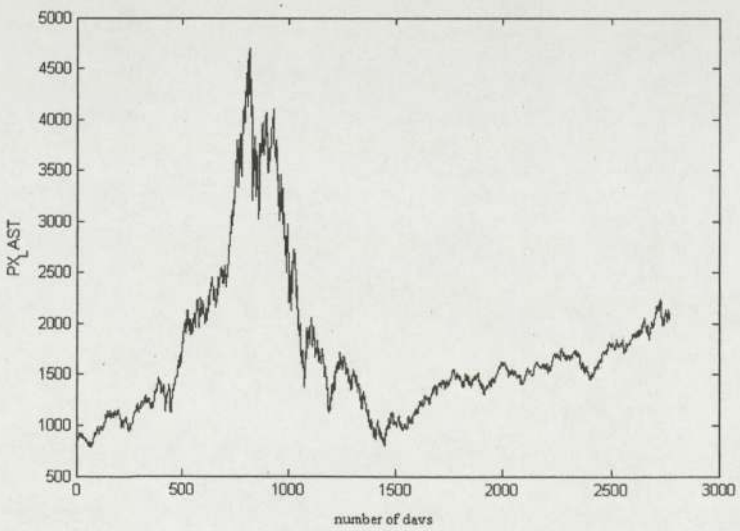

Figure 3.5: The NASDAQ-100 Index between 1997 and 2007.

All the experiments of this project have been run for these five indices.

3.2 Autocorrelation and partial autocorrelation

In this section we explore the correlation structure of the time series in order to see if an autoregressive model could be appropriate for the data. The autocorrelation helps to find repeating patterns in a process, such as the presence of periodic components. By definition, the total autocorrelation coefficient for a given time series X_t is [27]:

$$p_X(h) = \frac{\gamma_X(h)}{\gamma_X(0)} = Corr(X_{t+h}, X_t), \tag{3.1}$$

for all t and for all h where h is the delay.

It means that the correlation is computed between the original time series T and T delayed by a lag of size h. Autocorrelograms can be built by applying this formula to the different indices for large numbers of lags.

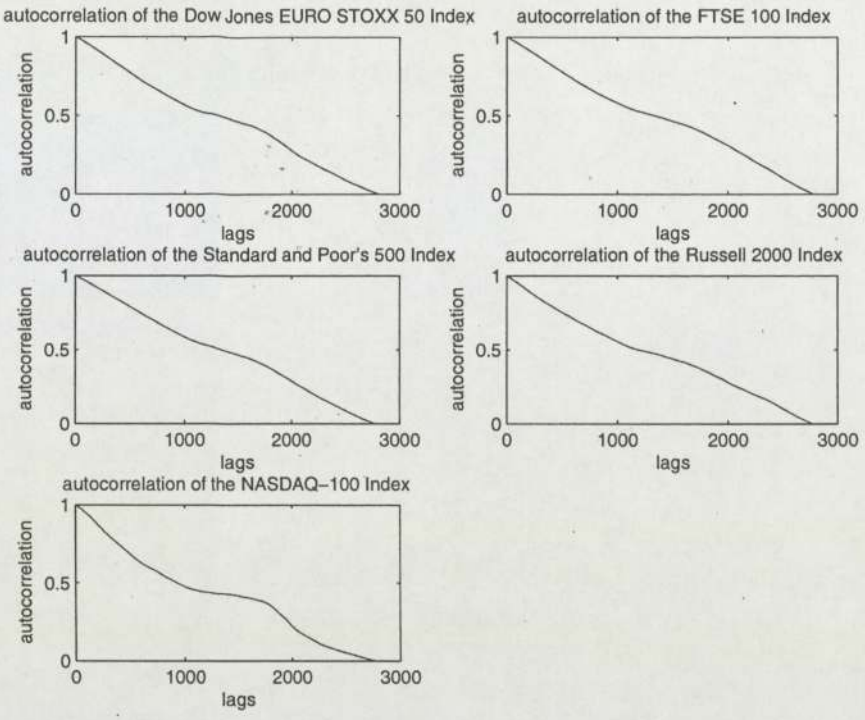

Figure 3.6: Autocorrelation of the 5 sets of data.

We notice that the autocorrelation is decreasing slowly on all the charts. Apparently the data remained autocorrelated all over the different lags. Therefore it is not possible to apply a Moving Average (MA(q)) model because the autocorrelation does not remain around 0 after a certain delay. However, these results must be interpreted carefully because it is likely that the time series are non-stationary.

In theory, partial autocorrelation shows if one can apply an AutoRegressive (AR(p)) model to a stationary time series. The partial autocorrelation is used to measure the relation between observations k steps apart after removing the correlation between observations 1, 2, ..., $k-1$ steps apart. By definition, the partial autocorrelation at lag k is the autocorrelation between X_t and X_{t-k} that is not accounted for by lags 1 to $k-1$ [27].

The partial autocorrelation between X_1 and X_k is given by the following formula:

$$r_{X_2,\ldots,X_{k-1}}(X_1, X_K) = corr(X_1 - P_{X_2,\ldots,X_{k-1}}(X1), X_k - P_{X_2,\ldots,X_{k-1}}(Xk)) \qquad (3.2)$$

corr = coefficient of classic correlation

$P_{X_2,...,X_{n-1}}(X_1)$ = projection of X_1 in the vector space generated by $X_2, ..., X_{k-1}$.

So if the partial autocorrelation is computed for a window of one year or one month, the partial correlation decreases very quickly after oscillating around 0 for all the indices.

Considering a period of one week, we obtained the graphs in Figure 3.7 :

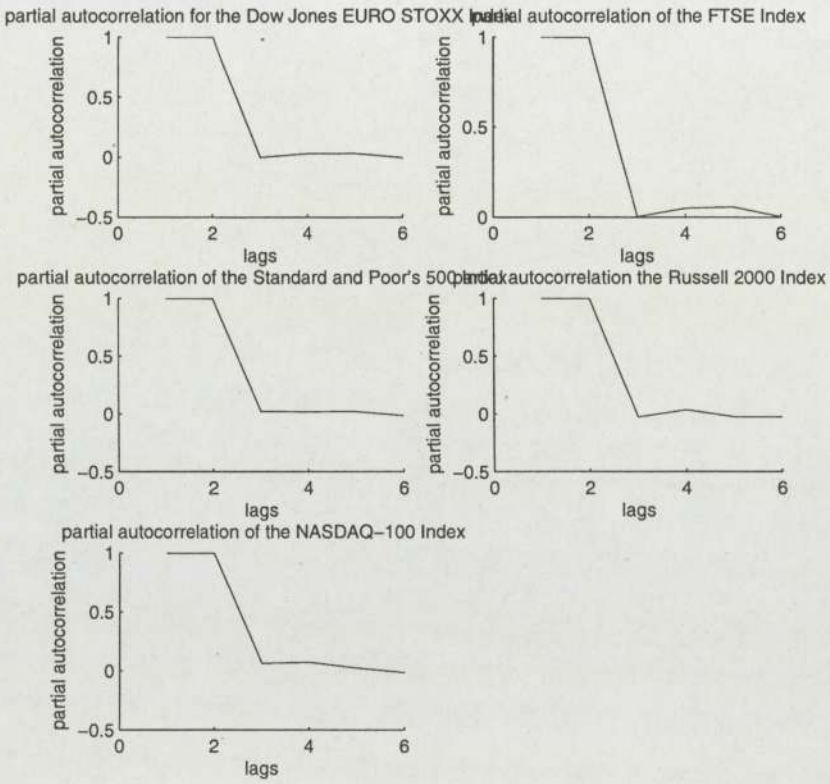

Figure 3.7: Partial Autocorrelation.

These graphs represent the evolution of the partial autocorrelation for the five indices for five lags, so the equivalent of a trading week. After the first day, the partial autocorrelation is oscillating around 0, which means there is no $AR(p)$ model that would be appropriate for the time series for $p > 1$. Of course, a random walk model is $AR(1)$. There does not appear to be any significant autocorrelation in any of the stock index time series.

3.3 Unit Root Tests

In this section, the stationarity of the stock indices is studied. A time series is weakly stationary if its mean value and covariances are time invariant: in the finance literature, the weak stationarity of asset return series is often assumed.
However, most stock price time series are not stationary and also very noisy. On the index plots, there appears to be no long-run average level about which the series evolve: this is evidence of non-stationarity. Statistical tests exist to prove that a time series is stationary or not: for example, the Dickey Fuller Test (1979) [3] augmented Dickey Fuller Test (1981) [9] and Phillips and Perron Test(1988). All these tests state on the modelling of the series by an $AR(p)$ model: a series is not stationary if when fitting the following model, one finds out that α is not significantly different from 0:

$$y_t = \beta + \alpha y_{t-1} + \epsilon_t. \tag{3.3}$$

It means that the system has a unit root.

To apply the Dickey Fuller Test, equation (3.3) is transformed [27]:

$$y_t - y_{t-1} = \beta + \alpha y_{t-1} - y_{t-1} + \epsilon_t, \tag{3.4}$$

$$\Delta y_t = \beta + (\alpha - 1) y_{t-1} + \epsilon_t. \tag{3.5}$$

Thus the null hypothesis H_0 is defined:

H_0 = the series is not stationary and there is a unit root $\alpha = 1$ or $(\alpha - 1) = 0$
So if t (test) $> t$ (table of Dickey-Fuller), H_0 is accepted and the series is non stationary. This test is run for three different cases to find out if the series is stationary or not: without drift, with a drift and with a time trend. If H_0 is accepted for one of them, the series is not stationary, y_t is differenced and the Dickey Fuller (DF) test is performed until H_0 is rejected. Most of the time, financial time series are non-stationary mainly because there is no fixed level for the prices.

The augmented Dickey-Fuller test is the same as the simple Dickey-Fuller test but is applied to another model which takes into account the autocorrelation of the

differenced time series with a correction on the delayed values:

$$y_t = \beta + \alpha y_{t-1} + \sum_{i=1}^{p} (\gamma_i \Delta y_{t-i}) + \epsilon_t. \tag{3.6}$$

This test is used to prove that the indices are non-stationary time series. By using the GRETL software [1], the following results were obtained:

	SX5E	UKX	SPX	RTY	NDX
result	-0.0019791	-0.0026511	-0.0027134	-0.00109	-0.00219
t val	-1.88234	-1.97087	-2.142	-1.15201	-1.78695
H_0 accepted	yes	yes	yes	yes	yes

Table 3.1: Augmented Dickey Fuller Test.

According to these results, the null hypothesis is accepted for all the indices. It is now assumed that the index series are non-stationary.

3.4 Returns

Most financial studies consider the *returns* of assets instead of prices. Two main reasons are invoked for using returns by Campbell, Lo and MacKinlay(1997) [36]: first, the returns series is a complete summary of the investment opportunity which are easier to manipulate. Indeed the returns usually have more attractive statistical properties, such as stationarity, than price series. As it is observed about the indices, the prices could be non stationary and the returns could be stationary for a same series.

In order to avoid some problems due to non-stationarity, it is often useful to consider the times series of returns or logarithms of returns instead of the values themselves. In the following charts, the time series of returns (called "one period single return" [27] or shorter "simple return") is computed. It means that the return of the assets are calculated between the time $t-1$ and t. In this case, t is the current day and $t-1$ the previous day.

If P_t is the asset price at time t and R_t is the asset return at time t,

$$R_t = \frac{P_t}{P_{t-1}} \tag{3.7}$$

By applying equation (3.7) to the index datasets, the following graphs are obtained:

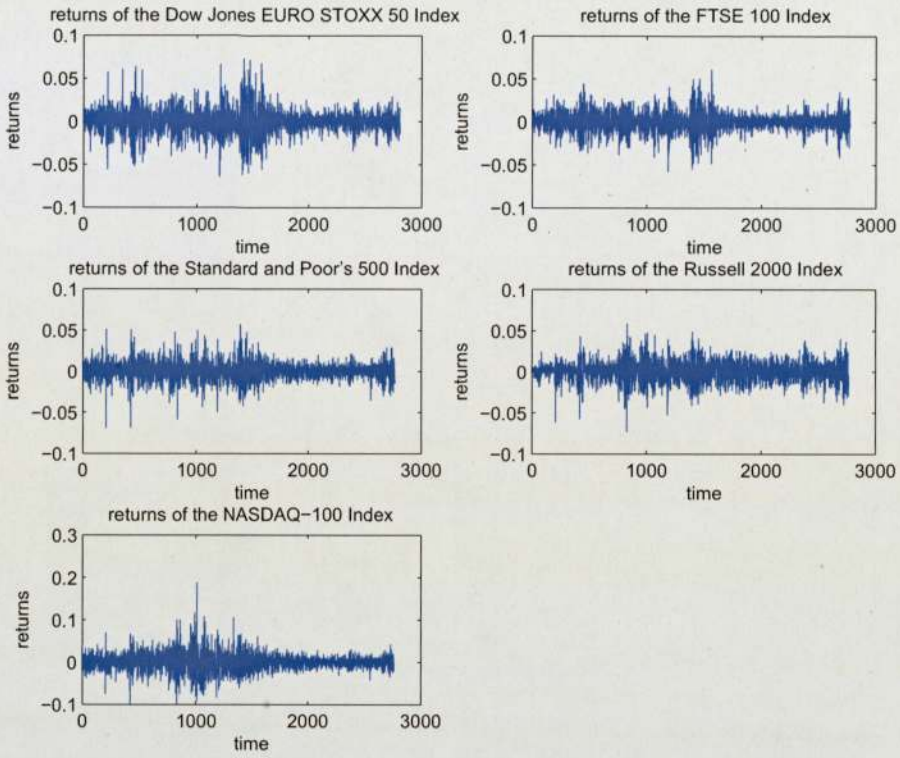

Figure 3.8: Returns time series from the original data.

It is noticeable that for all the graphs, the curves seem to be more regular than the prices and there are sometimes quite big variation upon one day. They look quite stable over the time and stationary.

So, as before, the autocorrelations is computed for the returns to investigate if it confirms our intuition.

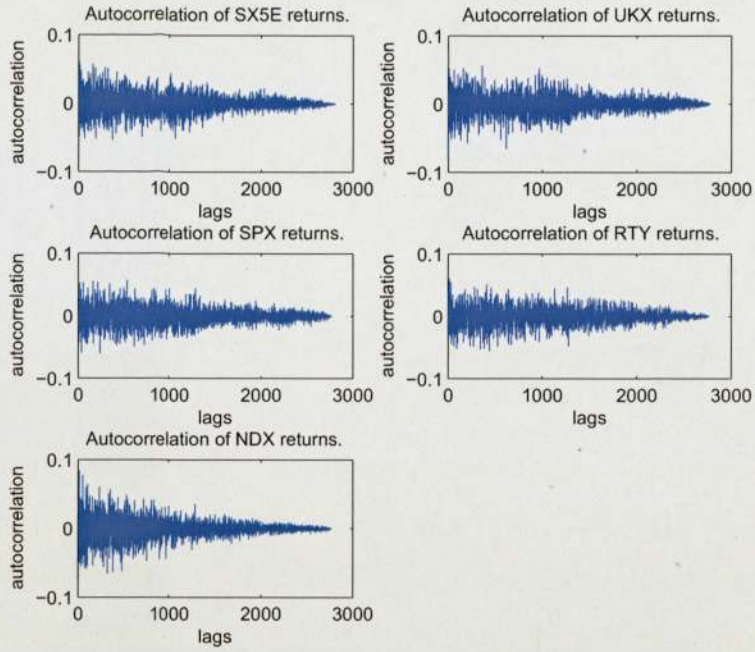

Figure 3.9: Autocorrelation of returns.

The graphs obtained are totally different than for the price series. Indeed, after the first lag, the autocorrelation oscillates around 0, but the absolute value decays slowly even though there is no serial correlation. We chose to stop studying the returns curve at this point. Indeed the Dupire process we will apply to the data includes defining levels, crossing points and distances between the strings. Hence a return is either positive or negative and it is defined by the current price divided by the last price. If levels are defined, they separate different sets of return values. Supposing a positive level is reached by the return path, the interpretation in terms of trading rule is not obvious to determine. Indeed if the return is positive, the today's price will be higher than yesterday's price but maybe lower than the price on the day the asset was bought. Consequently we would need to compute simulation of returns for a large number of cases. Moreover two or three levels will be enough to determine when the expected return is either positive or negative. It is clearly not the process described by Dupire and, as it was explained before, some studies already predict the sign of the expected return to know the direction of the next move. So we consider now only the index values to achieve the model.

3.5 Distributions

In this section, the distributional properties of the indices daily evolution are studied. It is interesting to understand the behaviour of the series over time to help determine the level size in the Dupire model.

A classical statistical normality test is run on the daily evolution of prices and gave the following plots for the SX5E index.

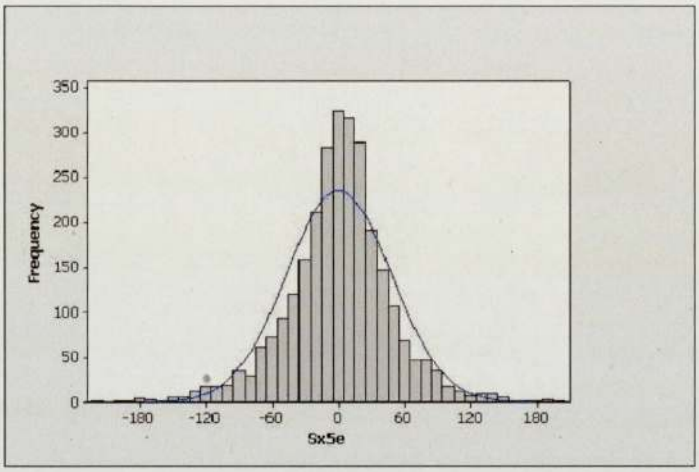

Figure 3.10: Frequency of the daily evolution and comparison with the normal distribution.

This plot shows an histogram of daily evolution and the associated normal curves. It is noticed that the curve does not seem to fit well to the data. The peak and the tails of the histogram are above the normal density curve; this is a fat-tailed distribution which is very common with financial time series. Therefore a Anderson-Darling normality test was run which is defined as [2]:

H_0: the data follow a normal distribution.

H_a: the data do not follow a normal distribution.

H_0 is rejected if the "p-value" calculated is less than the significance level

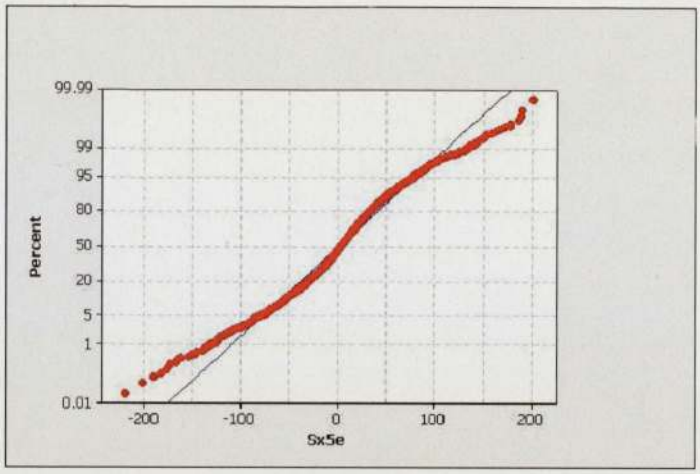

Figure 3.11: Normal test for SX5E.

According to the calculations, H_0 is rejected because $AD = 21.9$ and $P - Value < 0.005$. The distribution is not normal and the same result was obtained with the four other indices. The data distributions are more fat-tailed than a normal distribution. It means in particular that very extreme variations happen more often than the normal distribution predicts. This is what Mandelbrot called the Noah Effect [2]. To confirm this, the skewness and the kurtosis of the data were calculated.

Skewness is a measure of asymmetry: a negative value indicates skewness to the left and a positive value indicates skewness to the right.

Kurtosis is a fourth-moment measure of the shape (peakedness) of a density. It has the value 3 for a normal distribution. A positive value typically indicates that the distribution has a sharper peak than the normal distribution and a negative value indicates that the distribution has a flatter peak than the normal distribution [27].

	mean	std	skewness	kurtosis
SX5E	0.8956	47.456	-0.15	1.89
NDX	0.444	47.961	0.11	10.95
RTY	0.142	6.797	-0.25	1.7
SPX	0.257	13.102	-0.13	2.69
UKX	0.849	59.98	-0.23	1.58

Table 3.2: Distribution moments.

For all indices the skewness is quite different from 0 and the kurtosis is bigger than

for a normal distribution, especially in the case of the NASDAQ indices. One can conclude than unexpected extreme variations happen more often for these indices.
To implement the model described in the last chapter, the size of the levels must be defined. As the optimal choice of level depends on the nature of the prediction task, their size is chosen depending on the shapes of distributions. To predict the movement on the following day, a smaller threshold than the one for a two weeks horizon is used. Looking at the data repartition, the optimum levels must be a percentage of the standard deviation like 5%, 10% or 15%. For the experiments, this value is determined according to the expected degree of precision.

3.6 Conclusion

This chapter confirmed that the data studied have the well known features of financial time series: non-stationarity, non-normal distribution, and small partial autocorrelation. Therefore it is not possible to apply a standard time series model to the prices and it is one reason why Dupire's process may be of interest. Furthermore, the statistical properties of the data permit us to define one parameter of Dupire's model, the size of the levels.

Chapter 4

String Analysis

In this chapter, it is assumed that the sequences of binary strings have already been built. The purpose of this chapter is to study the strings' statistical properties.

4.1 String length

In this section we apply several methods in order to determine the optimal length of string for prediction.

4.1.1 Entropy measure

Entropy measures the degree of disorder of a thermodynamical system (Boltzmann entropy) or the mean quantity of information generated by the data (Shannon entropy). This tool applied to the structure of sequences was introduced by Shannon in 1951 in [10]. Furthermore, his concept was also applied to many other languages and other fields like biosequences, information processing, medical diagnosis and physics. Kolmogorov [19] investigated dynamic processes and used the Shannon entropy to calculate the algorithmic Kolmogorov complexity and established what is now usually called the Kolmogorov-Sinai entropy. Both measures are frequently used to investigate irregular time series which have high correlation order.

By definition, the formula of the Shannon entropy is, for a sequence of discrete observations:

$$I_{Window} = -k \sum_{i=1}^{N} p_i \log_2 p_i, \tag{4.1}$$

$$p_i = \frac{nb(occurrences(i))}{nbtotal(windows)}, \tag{4.2}$$

and $k > 0$, in the case of bits, $k = 1$ and $\ln = \log_2$.

Therefore, the entropy of the data sets is calculated in order to determine the optimum size of the window, which is equal to the length of the strings. The entropy is graphed for different window sizes for the different indices and for several level sizes. It was noticed that the results were similar for all the experiments; the following curves were obtained with the SX5E index.

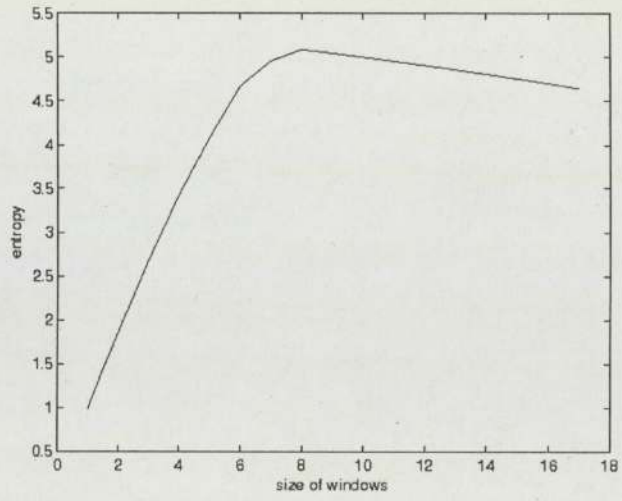

Figure 4.1: Graph of entropy as a function of the window size for 10 levels.

In this case, it is obvious that the curve reaches a maximum for strings of length 8 and decreases after the peak. So in this case windows of size 8 contain the maximum information or the maximum randomness. If we are looking at smaller levels, the general behaviour of the plot is the same but the maximum is reached for strings which are a little bit longer.

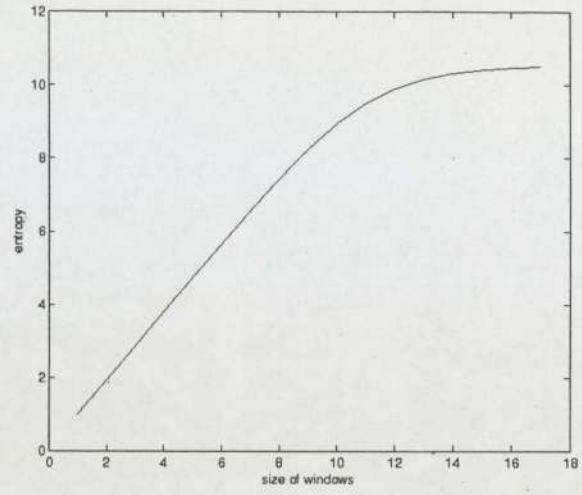

Figure 4.2: Graph of entropy as a function of the window size for 200 levels.

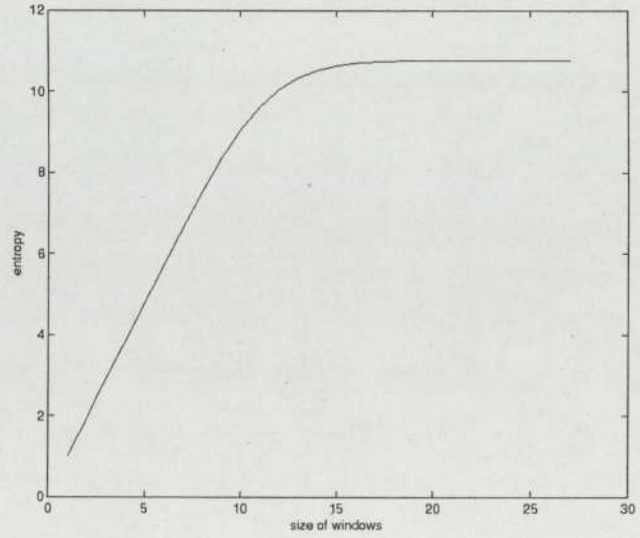

Figure 4.3: Graph of entropy as a function of the window size for levels of size 10.

For very small levels, the entropy reaches a near-maximum for windows of size 12 and start to decrease after 22. It means that strings of size 12 hold a sufficient quantity of information, very close to the maximum of information. With this length, the randomness reaches the point when the strings represented in the dataset appear with roughly equal probability. For greater window sizes there is an excedent of information which decreases the randomness. The experiments performed for the other indices and

other level sizes confirm empirically these results, and so we concluded that 12 may be a good value for the string length although this was larger than we had anticipated. Indeed a window of 12 bits means that there are $2^{12} = 4096$ different possible strings. As there are about 1000 windows after pre-processing the dataset, it means that it will be probably unlikely to find more than 2 occurrences of a given string of that length.

4.1.2 Conditional entropy

Assuming the word sequences of length k have already been built, it is possible to compute the *conditional entropy* which measures the uncertainty of predicting the next letter. This is actually the difference of the Shannon entropy computed for words of size $k+1$ and words of size k [23].

$$h_k = H_{k+1} - H_k. \tag{4.3}$$

The conditional entropy was plotted for the different indices. The following curve is obtained for the NDX index. It was noticed that the graphs have the same shape for the other indices.

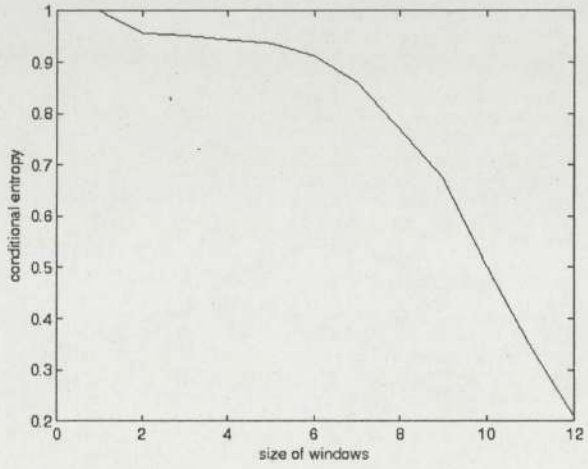

Figure 4.4: Graph of the conditional entropy as a function of the window size for 200 levels.

The curves decrease slowly at the beginning and faster for windows of size bigger than 6. The existence of long correlations is expressed by long decreasing tails of

the conditional entropies. Moreover *"beyond $k = 5$ or $k = 6$ the calculation of the conditional entropy is not reliable due to large statistical error"* [17]. It is concluded from these experiments that 6 could be a good length for the strings. Indeed, 6 means that there are $2^6 = 64$ different possible strings. Our view was that this was better than a length of 12 because it would be easier to create a predictive model that generalised well with smaller pattern space.

4.1.3 Singular Value Decomposition

A singular value decomposition of the data was done in order to determine by another method the optimum length of the strings. The goal of this experiment was to look for a threshold for the singular values which would indicate the optimum size of window. Unfortunately, this experiment gave no decisive result: for all the indices and for all the sizes of levels, the singular values were decreasing too quickly after the first one. Here are the results for the SX5E index:

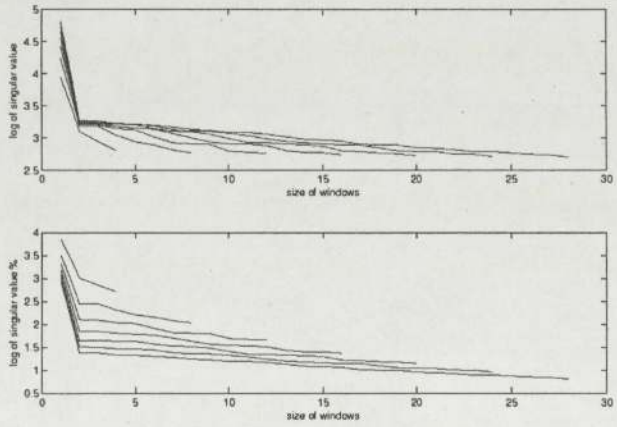

Figure 4.5: 1. Singular value decomposition for SX5E index.2. Percentage of singular value decomposition for SX5E index.

4.2 String statistics

By creating the binary strings, a time series of words is obtained. The same statistics used for the price sequences can be computed for the string time series in

order to understand their properties better.

4.2.1 Autocorrelation

In order to discover if the strings is correlated, the autocorrelation was computed for the string time series. To apply the usual formula, it is necessary to transform each string to its corresponding value in the decimal system. So each word is represented by a number lying between 0 and $2^{sw} - 1$ where sw is the length of the strings (i.e. size of windows). The most recent value is mapped to the largest power of 2. The autocorrelation was computed for windows of size 6 and 12 following the conclusions of the last section. Similar results were obtained for all the plots so only the curves graphed for the SPX index are shown.

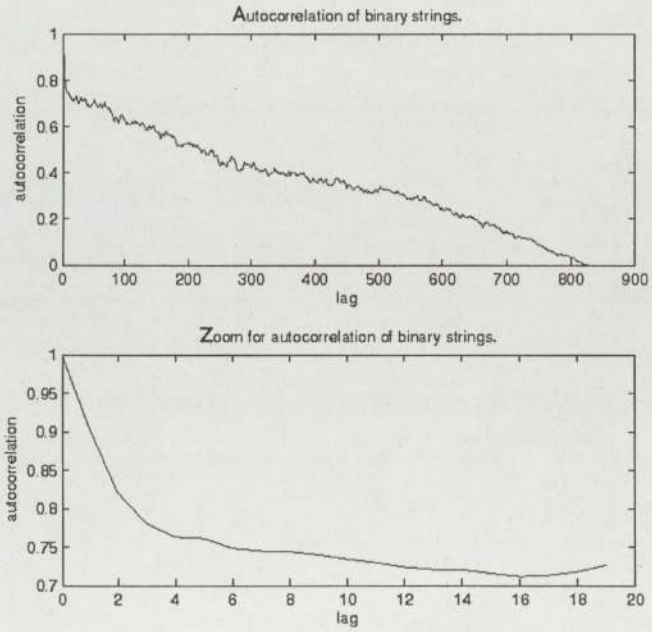

Figure 4.6: Autocorrelation for SPX index with words of length 12.

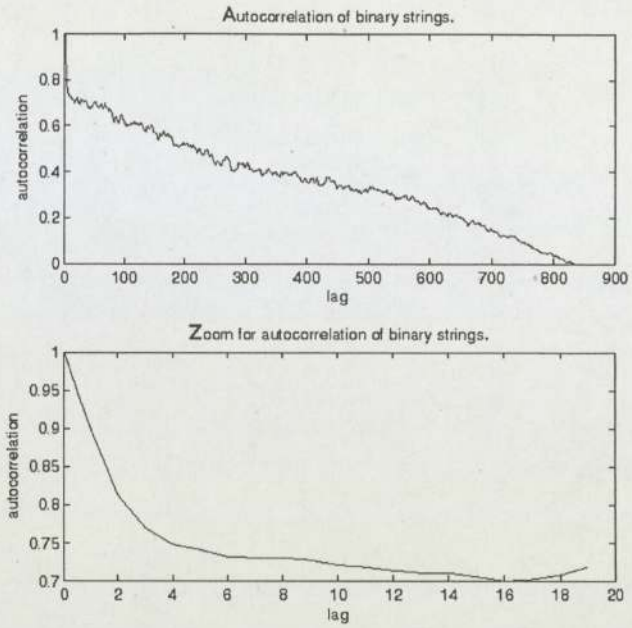

Figure 4.7: Autocorrelation for SPX index with words of length 6.

The strings seem to stay strongly correlated over all lags but there is a big decrease at the beginning. That is the reason why a zoomed plot of the autocorrelation on the smallest lags is shown. It appears that until the third lag the curve is decreasing quickly. Thus, the string times series are significantly more autocorrelated during the three first lags than afterwards. This property is true for words of both size 6 and size 12. Moreover the values of the autocorrelations are similar whatever the string length. It is concluded that here again a Moving Average (MA) model is not appropriate.

As for the price time series, the partial autocorrelation was computed for the string time series. The experiments were run for words of size 6 and 12 by transforming the binary strings to a decimal number in order to apply the usual partial autocorrelation formula. The same type of plots were obtained for all the sets of data. Figures 4.8 and 4.9 present the partial autocorrelation graphs of the SX5E index:

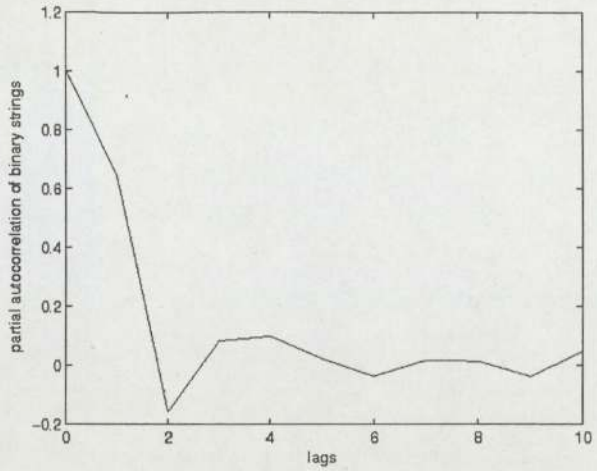

Figure 4.8: Partial autocorrelation for SX5E index with words of length 12.

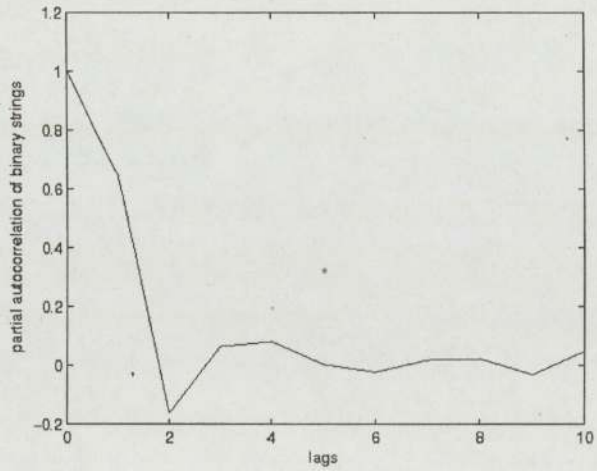

Figure 4.9: Partial autocorrelation for UKX index with words of length 6.

After the first lag, the partial autocorrelation oscillates around 0 although the variations are bigger than those observed with the original indices values. The curve indicates that the amplitude can fluctuate about 10% around 0. Thus, there is a very weak coefficient of partial autocorrelation between the different lags. It is concluded that it is not possible to find a good AR model for the string time series.

4.2.2 Distribution

It is interesting to look at the string distribution, since this could indicate if some sequences happen more often than others, if they following a particular distribution or if there is a pattern which has a high or low frequency. In this aim, the string distributions are graphed for all the indices and sequences of length 6 and 12. The following figures present the curves obtained for NDX and UKX with strings of size 6.

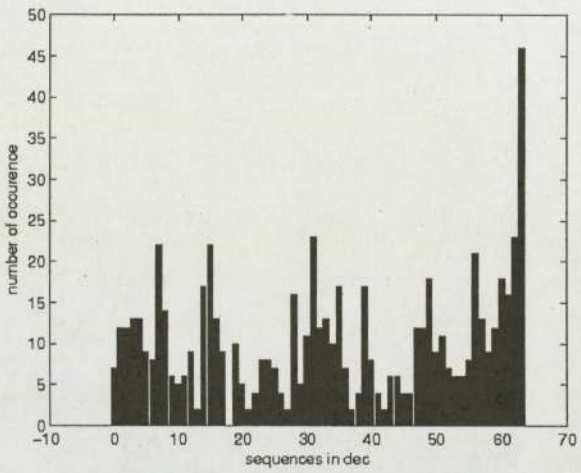

Figure 4.10: Strings histogram for the NDX index with words of length 6.

The histogram does not seem to follow a particular distribution. It is noticed that the word consisting entirely of ones has the biggest frequency, almost double the number of any other word. There are some strings which are totally absent from the distribution. This shape is similar for all the indices, but the UKX histogram contains more peaks, consequently, the peak of "111111" is a little bit smaller than the others.

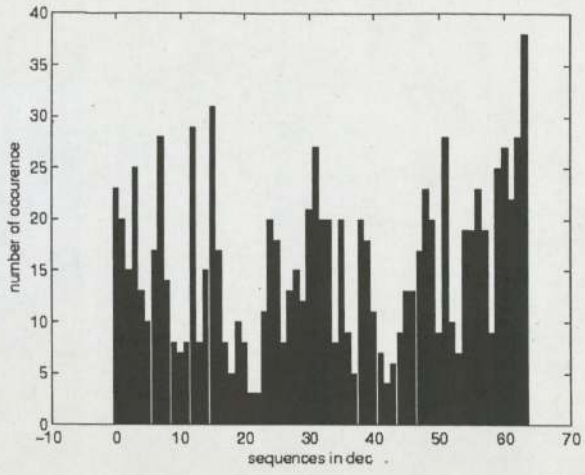

Figure 4.11: String histogram for the UKX index with words of length 6.

This is the plot obtained for NDX index and strings of size 12:

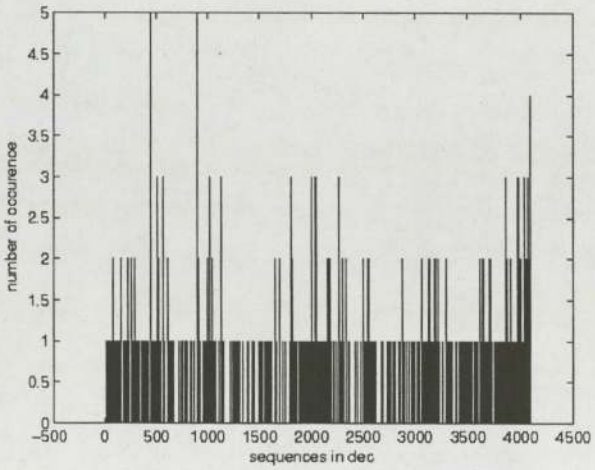

Figure 4.12: String histogram for the NDX index with words of length 12.

In general these graphs generated with longer words are really undersampled; the biggest occurrence which is reached is 3 and there are a lot of strings which are not present. According to the maximum entropy theory, the probabilities for all the strings are almost equal, there is practically only one occurrence of each possible sequence. As the model computes the distance between the strings, the fact that we do not have the list of all the words in the distribution should not pose any difficulty.

To conclude, it is not possible to fit these histograms with a classical distribution such as the normal distribution.

4.3 String Entropy

The next step was an investigation of the dataset size needed to create a model that generalises. To do this, we calculated the Kolmogorov entropy function of the number of strings. The aim was to find out if the entropy has a maximum or not; if such a point existed, it would mean that the corresponding number of strings contains the maximum of past information and that using more strings to train the model will be a waste of time. The following graphs are obtained for the RTY index with binary words of size 6 and 12:

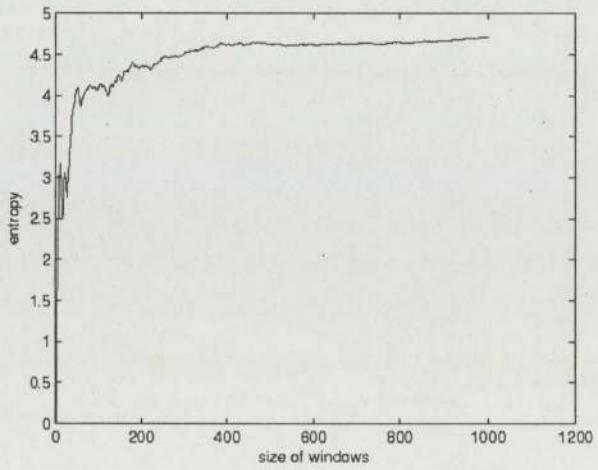

Figure 4.13: Kolmogorov entropy for SPX index with words of length 6.

It is observed that the curve was irregular especially at the beginning where there are large fluctuations, while for sets of more than 300 strings, the curve continues to increase but very weakly. Thus we concluded that to obtain enough information, at least 300 words are required to train the model. This big number would probably contain enough long-range memory to achieve good predictions.

Here, the path was considerably more regular, especially around the inflexion point, but even if the entropy increased quite slowly after this point, it still rose steadily. We

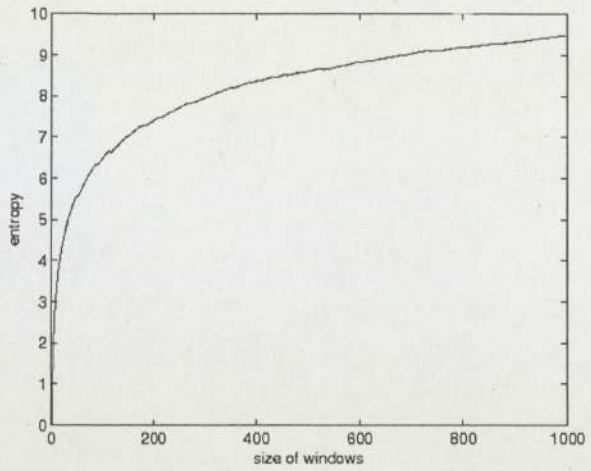

Figure 4.14: Kolmgorov entropy for SPX index with words of length 12.

hypothesised that, as the size of the strings was bigger than previously, the graph would probably have the same behaviour if it was possible to pursue the experiments with more data. Recall that we showed in Section 4.2.2 that the data were undersampled for this size of string and more data would be required to reach valid results.

As it could be interesting to know the proportion of digits 1 and 0 in the time series, this information is summarized in the following table:

%	SX5E	NDX	SPX	RTY	UKX
0	47.15	45.68	46.83	44.63	47.15
1	52.85	54.32	53.17	55.37	52.85

Table 4.1: Bit distribution.

It is noticeable that for all the series there are more 1s than 0s, which means that the market prices have a higher probability of increasing (by at least the amount between levels) than decreasing. That could explain the reason why if a Buy & Hold strategy is applied during a long time, there is a good chance of making a profit.

The experiments run with the other indices showed the same feature so we concluded that a data set containing about 300 words should be sufficient train the model. In the next chapter, the model will be implemented.

Chapter 5

Model implementation

This chapter investigates the implementation of the model introduced by Dupire and compares it with other models, such as neural networks. In order to optimize the process, the right values of the Dupire's model parameters need to be determined [6]:

$$p = \frac{\sum e^{-\lambda d(X,Y_i)} * p_i}{\sum e^{-\lambda d(X,Y_i)}}, \tag{5.1}$$

and

$$d(X,Y) = \sum_{j=1}^{N} \alpha^{N-j} * (X_j - Y_j)^2, 0 < \alpha < 1 \tag{5.2}$$

where $p_i = 0$ or $p_i = 1$ is the digit following Y_i.

The two parameters to be established to use this model are α and λ. We tried to determine them using two techniques: cross validation and maximum likelihood.

5.1 Cross-validation

Cross-validation is a commonly used empirical model evaluation method. For instance, it has been used to compare neural networks with different architectures in order to determine which has the best performance on new data to avoid overfitting. In this application, we will compare the performance of different models by evaluating the error function on different sets than those used for the training. There are several

algorithms for this purpose: hold-out, leave-one-out and K-fold cross validation. The differences between these methods lie in the way the sets are defined before computing the error for each of them. We chose to apply the simplest cross-validation, the hold-out method. The data were separated in three independent sets, called the training set, validation set and test set with the respective proportions: $\frac{1}{2}, \frac{1}{3}, \frac{1}{6}$. These sets were selected as blocks of data keeping the time order intact. A range of models with different values of $0 < \alpha < 1$ and $0 < \lambda$ were trained using the training set. The performance of the models was compared by evaluating the error function using the validation test. The ordered pair (α, λ) minimizing the error function was selected. To get an unbiased measure of generalisation, the performance of the selected network was measured on the test set and compared with the error for a benchmark model (such as random walk) [18]. The random walk process is defined by the following model,$p_i = p_{i-1}$ (where p_i is the i^{st} move): the next move is the same as the last one.

The error function is defined by the following formula:

$$E = \sqrt{\frac{\sum(t_i - p_i)^2}{\sum(\bar{t} - p_i)^2}}, \tag{5.3}$$

where t_i is the i^{st} bit, p_i the probability for the i^{st} bit and $\bar{t}$ the mean. A value of 1 indicates a model that is no better than predicting the mean outcome, while a value of 0 means perfect predictions.

This method was run for all the indices with strings of size 6 and 12. It gave these results for the SX5E index:

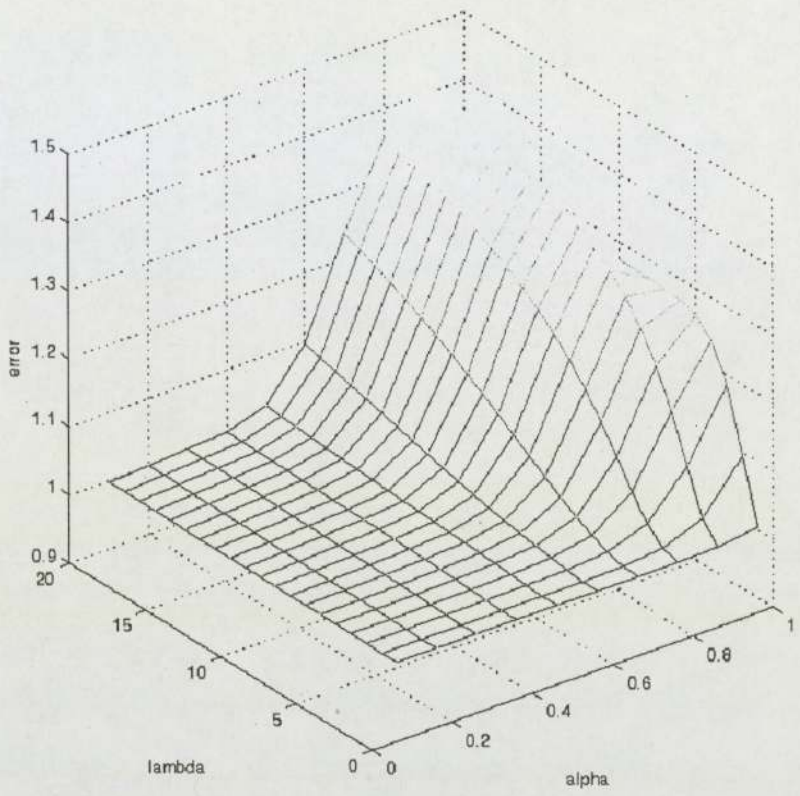

Figure 5.1: Error for the validation test of SX5E index with strings of size 12.

We are looking for a minimum of the error surface. However, there is no sharp dip around the minimum, so determining the exact value is difficult; it was achieved by zooming in the graph. This also means that the value of parameters need not be fixed precisely, as a small variation will not have a significant effect on the predictive accuracy. Moreover the minimum is not necessarily unique, as it could be reached at several points.

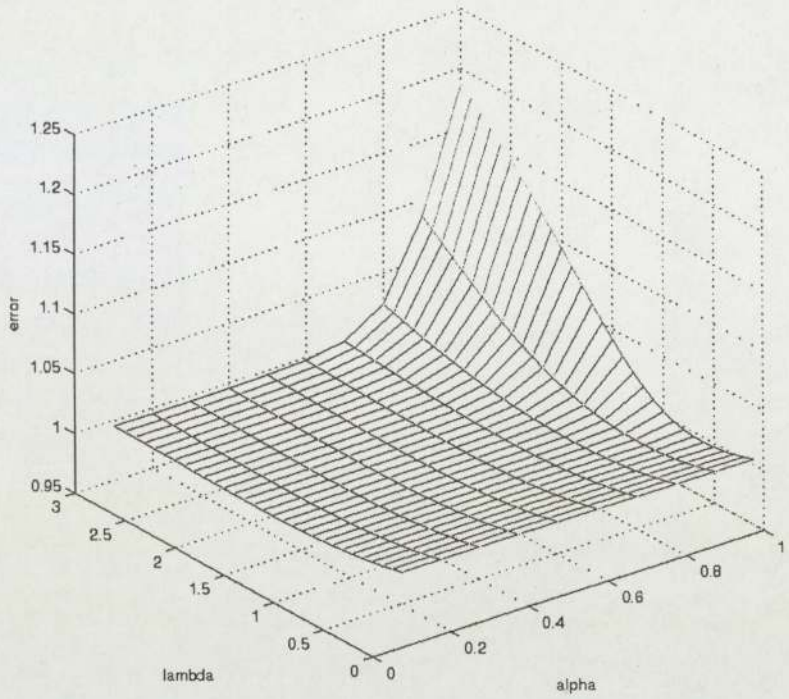

Figure 5.2: Error for the validation test of SX5E index (zoom) with strings of size 12.

In this example, the error was minimum for $\alpha = 0.6$ and $\lambda = 1.5$. When the selected model is run on the test set, the following results were obtained (the letter in brackets indicates the name of the corresponding column in the following array):

- 35 (e1) predictions were false out of 97 examples (card), which means the false prediction rate is around 36% (E1).
- The normalized error on the validation test was 0.9550 so it was a little better than the average (e2).
- If the random walk error (e3) is calculated, an error of 36%(E3) is obtained so exactly the same as the model.
- The trading rule was used as described before but only probabilities under 40% and bigger than 60% are considered. So the number of false predictions is evaluated if this probability threshold is considered. Then we find out that 12 prediction (e4) are false in the sample of 39 (cardrj) elements which correspond to an

error of 30.8% (E4) which is a little bit better than before.

The table 5.1 presents the summary of the experiments run for strings of size 12.

Index	SL	SW	a	l	card	e1	e2	e3	e4	cardrj	E1	E3	E4
SX5E	70	12	0.6	1.5	97	35	0.955	35	12	39	0.361	0.361	0.308
NDX	40	12	0.8	1.7	111	40	0.954	44	22	65	0.36	0.396	0.338
RTY	6	12	0.1	2.3	166	73	1.025	73	73	165	0.44	0.44	0.442
SPX	13	12	0.1	7	138	60	1.023	60	46	113	0.435	0.435	0.407
UKX	50	12	0.08	10	165	69	0.979	69	28	67	0.418	0.418	0.418

Table 5.1: Cross Validation results with strings of length 12 with:
***sw**: size of windows, a: alpha, **l**: lambda, **card**: cardinal of the test set, **e1**: number of false bits, **E1**: percentage of false bits, **e2**: normalized error, **e3**: random walk error, **E3**: percentage of random walk error, **e4**: number of false bits for the rejection method, **E4**: percentage false bits for the rejection method, **cardrj**: cardinal of the test set with the rejection method.*

The graphs have similar shapes for all the indices but the minimum is reached for a different ordered pair (α, λ) for each index.

Similar experiments have been run with strings of size 6 and the UKX index.

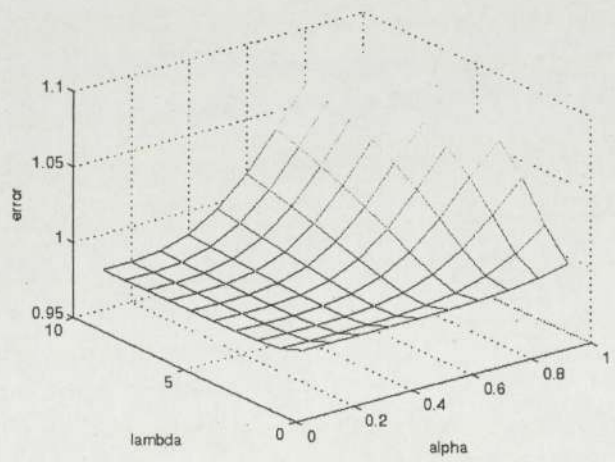

Figure 5.3: Error for the validation test of UKX index with strings of size 6.

The figure was very similar to the one obtained for strings of size 12. Here again the miminum seems to be in a broad flat area of the graph, so we had to enlarge the plot in order to find the right values. The error here is minimum for $\alpha = 0.11$ and $\lambda = 18$.

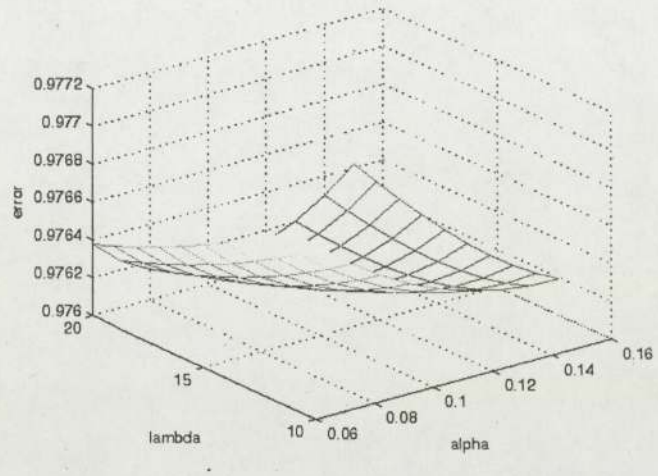

Figure 5.4: Error for the validation test of UKX index (zoom) with strings of size 6.

Table 5.2 presents results obtained with the test sets for several different indices.

Index	SL	SW	a	l	card	e1	e2	e3	e4	cardrj	E1	E3	E4
SX5E	60	6	0.08	1.8	113	45	0.976	45	23	68	0.398	0.398	0.338
NDX	40	6	1	1	111	40	0.97	45	19	57	0.36	0.405	0.333
RTY	6	6	0.01	2.8	166	74	1.034	74	74	166	0.446	0.446	0.446
SPX	13	6	0.08	7.1	138	61	1.02	61	31	83	0.442	0.442	0.373
UKX	50	6	0.11	18	165	69	0.98	69	26	68	0.418	0.418	0.382

Table 5.2: Cross Validation results with strings of length 6.

From these experiments, we can conclude that the error is not sensitive to the value of λ and α because the graphs look quite flat. Whatever the index and the string length, the error is similar to the one generated by a random walk process. That is not conclusive, even if the results obtained for the Nasdaq are better. Dupire defined the trading rule by considering only the probability higher than 60% and lower than 40%. When the number of false predictions is computed after deletion of probabilities close to 50%, the error is always smaller than achieved by a random walk. To find out the

performance of this method, we apply it to trading rules, it will be covered in the next chapter.

5.2 Maximum likelihood estimation

The maximum likelihood method was used to seek the optimum values for the parameters λ and α. The vector containing all the parameters is called w. For a sample of n data, the probability of observing all these points is the joint distribution $p(x_1, x_2, \ldots, x_n)$. The analysis is based on the assumption that the data is all independent and identically distributed, when the likelihood function for all n data points becomes [18]:

$$L(w) = \prod_{i=1}^{n} p(x_i|w). \tag{5.4}$$

The likelihood function gives the probability density of the observed data as a function of w. The value of w is selected by maximizing the likelihood of the model having generated the training data. In practice, it is often convenient to minimise the negative log-likelihood, which is seen as an error measure [18].

$$E(w) = -\ln L(w). \tag{5.5}$$

Combining this with equation the following expression is obtained:

$$E(w) = -\sum_{i=1}^{n} \ln p(x_i|w). \tag{5.6}$$

From this equation, the likelihood of the model is calculated, calling X the current string and p_i the bit following the current string.

$$P(p_x|w) = \frac{\sum e^{-\lambda d(X,Y_i)} * p_i}{\sum e^{-\lambda d(X,Y_i)}} = \pi(X). \tag{5.7}$$

This probability is computed for the training data:

$$\pi_j = \frac{\sum e^{-\lambda d(Y_j,Y_i)} * p_i}{\sum e^{-\lambda d(Y_j,Y_i)}}. \tag{5.8}$$

The classical binomial formula is used for the likelihood, the training data containing both the strings and their following bits:

$$L(\alpha, \lambda) = \prod_{j=1}^{n} \pi_j^{p_j} (1 - \pi_j)^{1-p_j}. \tag{5.9}$$

We can then compute the error as follows:

$$E(\alpha, \lambda) = -\sum_{j=1}^{m} (p_j \ln \pi_j + (1 - p_j) \ln(1 - \pi_j)). \tag{5.10}$$

The terms of the products are expanded:

$$\begin{aligned} E(\alpha, \lambda) = &-\sum_{j=1}^{m} p_j [\ln \sum_{i=1}^{n} \exp^{-\lambda d(Y_j, Y_i)} * p_i - \ln \sum_{i=1}^{n} \exp^{-\lambda d(Y_j, Y_i)}] \\ &- \textstyle\sum_{j=1}^{m} (1 - p_j) [\ln \sum_{i=1}^{n} \exp^{-\lambda d(Y_j, Y_i)} * (1 - p_i) - \ln \sum_{i=1}^{n} \exp^{-\lambda d(Y_j, Y_i)}]. \end{aligned} \tag{5.11}$$

By factorising the last expression, we obtain

$$\begin{aligned} E(\alpha, \lambda) = &-\sum_{j=1}^{m} p_j [\ln \sum_{i=1}^{n} \exp^{-\lambda d(Y_j, Y_i)} * p_i] \\ &- \textstyle\sum_{j=1}^{m} (1 - p_j) [\ln \sum_{i=1}^{n} \exp^{-\lambda d(Y_j, Y_i)} * (1 - p_i)] \\ &+ \textstyle\sum_{j=1}^{m} \ln \sum_{i=1}^{n} \exp^{-\lambda d(Y_j, Y_i)}. \end{aligned} \tag{5.12}$$

The parameter values that give the maximum likelihood cannot be found analytically, so it is necessary to use a non-linear optimisation algorithm. To apply such an algorithm, it is necessary to be able to compute the error gradient. By differentiating with respect to lambda, this is obtained:

$$\begin{aligned} \frac{\partial E(\alpha, \lambda)}{\partial \lambda} = &\sum_{j=1}^{m} p_j \frac{\sum_{i=1}^{n} \exp^{-\lambda d(Y_j, Y_i)} * d(Y_j, Y_i) * p_i}{\sum_{i=1}^{m} \exp^{-\lambda d(Y_j, Y_i)} * p_i} \\ &+ \textstyle\sum_{j=1}^{m} (1 - p_j) \dfrac{\sum_{i=1}^{n} \exp^{-\lambda d(Y_j, Y_i)} * d(Y_j, Y_i) * (1 - p_i)}{\sum_{i=1}^{n} \exp^{-\lambda d(Y_j, Y_i)} * (1 - p_i)} \\ &- \textstyle\sum_{j=1}^{m} \dfrac{\sum_{i=1}^{n} \exp^{-\lambda d(Y_j, Y_i)} * d(Y_j, Y_i)}{\sum_{i=1}^{m} \exp^{-\lambda d(Y_j, Y_i)}}. \end{aligned} \tag{5.13}$$

The derivative is calculated with respect to α, sw being the length of the strings. But there is a constraint on α in the model: $0 < \alpha < 1$. To avoid this issue, α is defined as a deterministic function of an unconstrained variable a:

$$\alpha = \frac{\exp^{-a}}{1 + \exp^{-a}}. \tag{5.14}$$

So we need to compute the derivative of the error function with respect to a. The first step is to differentiate the distance by a:

$$\frac{\delta d(Y_j, Y_i)}{\delta a} = \frac{\delta(\sum_{k=1}^{sw}(\frac{\exp^{-a}}{1+\exp^{-a}})^{sw-k}(Y_{jk} - Y_{ik})^2)}{\delta a}. \tag{5.15}$$

By computing the derivative:

$$\frac{\partial d(Y_j, Y_i)}{\partial a} = \sum_{k=1}^{sw-1}(sw-k)(\frac{-\exp^{-a}}{(1+\exp^{-a})^2})^{sw-k-1}(Y_{jk} - Y_{ik})^2, \tag{5.16}$$

the derivative of the error function is calculated

$$\begin{aligned}\frac{\partial E(\lambda, a)}{\partial a} = \sum_{j=1}^{m} p_j \frac{\sum_{i=1}^{n} \exp^{-\lambda d(Y_j,Y_i)} * \lambda * p_i * \sum_{k=1}^{sw-1}(sw-k))(\frac{-\exp^{-a}}{(1+\exp^{-a})^2})^{sw-k-1}(Y_{jk} - Y_{ik})^2}{\sum_{i=1}^{m} \exp^{-\lambda d(Y_j,Y_i)} * p_i} \\ + \sum_{j=1}^{m}(1-p_j)\frac{\sum_{i=1}^{n} \exp^{-\lambda d(Y_j,Y_i)} * \lambda * (1-p_i) * \sum_{k=1}^{sw-1}(sw-k)(\frac{-\exp^{-a}}{(1+\exp^{-a})^2})^{sw-k-1}(Y_{jk} - Y_{ik})^2}{\sum_{i=1}^{m} \exp^{-\lambda d(Y_j,Y_i)} * (1-p_i)} \\ - \sum_{j=1}^{m} \frac{\sum_{i=1}^{n} \exp^{-\lambda d(Y_j,Y_i)} * \lambda * \sum_{k=1}^{sw-1}(sw-k)(\frac{-\exp^{-a}}{(1+\exp^{-a})^2})^{sw-k-1}(Y_{jk} - Y_{ik})^2}{\sum_{i=1}^{m} \exp^{-\lambda d(Y_j,Y_i)}}\end{aligned} \tag{5.17}$$

Once all the terms of the gradient function had been determined, it was possible to implement the maximum likelihood algorithm using MATLAB and NETLAB [32]. The network is constructed in the NETLAB framework by creating the same type of data structures and functions. The model is called *'bst'* and the following functions were defined:

- *bstinit* constructs the network data structure with a training set, a target set (the bits following the strings in the training set) and the two parameters α and λ .
- *bstfwd* computes the predictions the model calculates from an input set.
- *bsterr* computes the log likelihood error for an input set.
- *bstgrad* computes the gradient over α and λ of the negative log likelihood error function.
- *bstpak* condenses the parameters α and λ in a vector w.
- *bstunpak* sets the parameters α and λ of the network from the vector w.

After implementing these functions, the "netopt" function in netlab was used with the option 'scg' (scaled conjugate gradients optimisation) algorithm to seek the minimum of the error function so the maximum of likelihood and then the optimum value for α and λ. The two last functions, *bstpak* and *bstunpak*, allow the new model to be used in conjunction with a general purpose non-linear optimisation algorithm in the NETLAB framework. The following values were obtained for the parameters:

Index	a	alpha	lambda
SX5E, 5, 6	−6.7861	0.9989	23.0565
NDX, 5, 6	−15.6674	0.9999	22.78
RTY, 5, 6	−9.4622	0.9999	23.4623
SPX, 5, 6	−11.0408	0.9999	13.9541
UKX, 5, 6	−43.3279	0.9999	16.8221

Table 5.3: Results of Maximum Likelihood.

The algorithm gives an estimate for α which is very close to its upper limit. However, the trend is the same for all the indices, an α almost equal to 1 and a λ around 20 which is much larger than the results obtained with cross-validation. The model given by these values will be evaluated in the next chapter.

5.3 Logistic regression

Logistic regression is a generalized linear model used when the target variable is a binary discrete variable, which corresponds to a single-layer perceptron or single-layer artificial neural network. These models calculate a linear combination of the input variables, in which the coefficient are the parameters, followed by an activation function appropriate to the type of data being modelled [18].

$$a_j = \sum w_{ji} x_i + b_j, \tag{5.18}$$

with a_j associated to each output.

Then a_j are transformed by an activation function which is monotonic. This function is

the logistic sigmoidal activation function applied to each of the outputs independently

$$y_j = \frac{1}{1 + \exp^{-a_j}}. \tag{5.19}$$

In our application, the input values are the strings converted to decimal numbers and the output is the following bit. So there is only one input neuron and one output neuron. This process permitted us to classify the strings into two classes, those followed by an "up" and those following by a "down". We implemented this model using the "glm" function of netlab with option "logistic". We recorded our results in a confusion matrix which is a convenient way of presenting the results of a classification model. They provide information on the model performance on each class. The rows represent the true class and the columns the predicted classes. Using the function "confmat" in netlab [32] we obtained for the SPX index with strings of size 12 the following matrix:

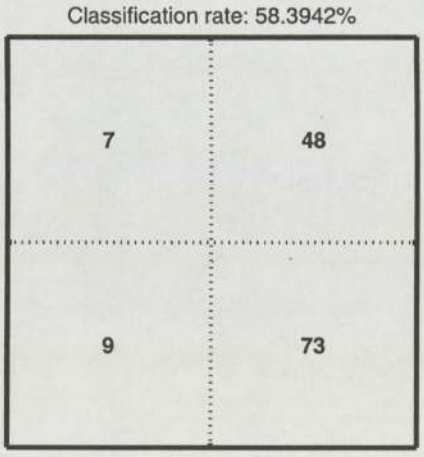

Figure 5.5: Confusion matrix from a logical regression for the SPX index with strings of size 12.

The classification rate was more than 58%, but we noticed the logistic regression tends to predict more "1" than "0". This is since, on average, there are more up than down moves in the data.

Moreover when we look at the other indices, sometimes the classification is empty for the bit"0". That confirmed our hypothesis, and if we run the classification on strings of size 6, we obtain the following matrix with the same index:

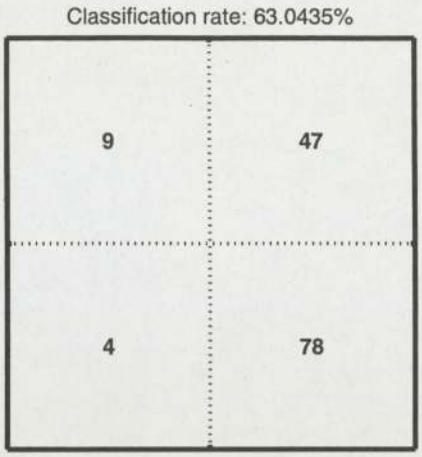

Figure 5.6: Confusion matrix from a logical regression for the SPX index with strings of size 6.

The rate here is better than for strings of size 12 but the main issue remains, the classification for the "up" is identical. In table 5.4, the resumed of the classifications rate are presented for all the indices with different sizes of string:

sw	SX5E	NDX	RTY	SPX	UKX
5	57.5221	57.6577	56.0241	61.5942	50
6	59.292	54.0541	57.2289	63.0435	55.1515
7	51.3274	50	57.2289	52.1739	55.1515
8	58.0457	50.9091	57.2289	54.3478	58.1818
9	54.4643	46.3636	57.2289	56.5217	53.9394
10	53.5714	55.4545	57.2289	59.854	55.1515
11	50.8929	58.1818	57.2289	59.854	55.1515
12	55.3571	58.1818	57.2289	58.642	54.878

Table 5.4: Classification accuracy as a function of the string length.

The results vary depending on the length of the strings, but the cardinal of the set predicting the bit '1' is always much bigger. The next step was to train a more complex logistic model, a multi-layer perceptron.

5.4 Multi-layer perceptron

The multi-layer perceptron is the most often used neural network. It follows the same principles as the simple perceptron but has more than one layer of adaptive

weights. The first layer activation variables a_j^1 are given by the linear combination of the inputs. They are then transformed by a logistic sigmoidal activation function. The output layer activation variables are a linear combination of the first layer activations. The final output value is a logistic sigmoid applied to the the output layer variables.

We used this model to classify the strings in two classes, the sequences following by a 1 and by a 0, so we have one input neuron which receives the string converted in decimal and one output neuron which predicts the following bit. However, the number of hidden neurons (first layer variables) in the perceptron is not known *a priori*. In order to determine it, we used cross validation again. The minimum of the error was determined for the validation set and then the generalisation performance of the network was evaluated on the remaining test data. The following figure is an example of the results obtained with the NDX index with string of size 12 and 6:

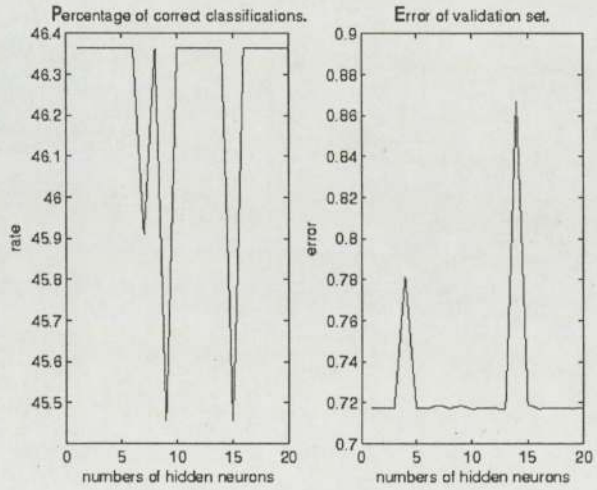

Figure 5.7: Cross validation results for an MLP with the NDX index and strings of size 12.

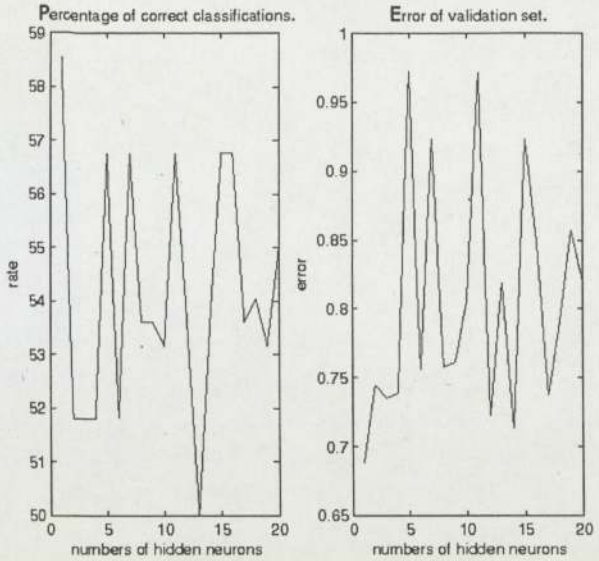

Figure 5.8: Cross validation results for an MLP with the NDX index and strings of size 6.

As the curves were very noisy, it was impossible to find a well-defined minimum of the error rate, and hence the optimum number of hidden neurons. Similar results are obtained with the other indices and size of strings. According to Figures 5.8 and 5.7, the optimum number of hidden neurons seems to be 4. The result is not really conclusive but nevertheless the confusion matrix was computed for this network architecture to give an idea of the model's performance.

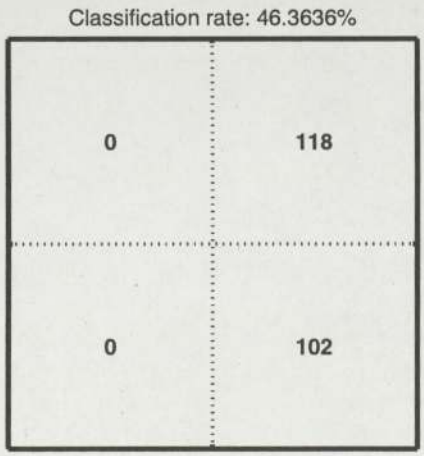

Figure 5.9: Confusion matrix for an MLP with the NDX index and strings of size 12.

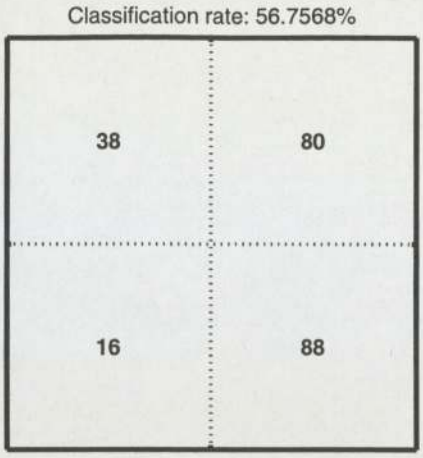

Figure 5.10: Confusion matrix for an MLP with the NDX index and strings of size 6.

The classification rates are no better than those for logistic regression. The matrix put in evidence the trend to class all the strings in the "up" class here again, but for strings of size 6, the prediction is more diversified between up and down. Finally, the results achieved with neural networks were not very convincing.

In the following chapter, the model is applied to effective trading rule.

Chapter 6

Trading rules

The goal of this application is to create a prediction model that can be used to trade profitably. Hence the results we obtained in the last chapter have to be evaluated through simulated trading. The model will be compared with a Buy & Hold strategy.

6.1 Performance evaluation by trading rules

6.1.1 Trading strategies

It is assumed that the probabilities for the next bit to be an "up" move are already computed by a predictive model. Then two thresholds which determine a high and a low limit have to be defined. When the probabilities are bigger or lower than these borders, we buy or sell some assets as described in the Dupire method [6] with limits of 40% and 60%. When the output probability is less than 40%, the trader sells the stocks because the model indicates that the prices are likely to decrease. On the contrary, if the model predicts an upward move with an probability of greater than 60%, the trader buys assets. Between these limits, the trader does not perform any action, he just holds the current position.

Several variants of this rule can be imagined. For instance, following the same principles, the high and low limits can be moved depending on the probability distribution. Another idea is to invest proportionally to the probabilities to have an up. Supposing

that the output is $x\% > limit_{up}$, the trader buys assets for $x\%$ of our capital. Logically, if $x\% < limit_{down}$, he will sell stocks for $100\% - x\%$ of the capital.

Once the trading rule is defined, its performance is evaluated. There are several ways to measure performance: the first is to compare the profits generated by our strategy to those obtained using a Buy & Hold strategy [24]. It is a long term investment strategy based on the concept that, in the long run, financial markets give a good rate of return despite periods of price volatility or decline. The strategy consists of buying at the beginning of the period as much stock as the trader can afford and to save it until the end.

To compare the performance of the Binary Strings strategy and Buy & Hold, we computed the annualized returns and the annualized volatility. Indeed it was not sufficient to simply look at the returns because that does not account for the risks in trading. The annualized return for a period T (measured in years) is given by:

$$R_T = \left(\frac{S_T}{S_0} - 1\right)/T, \tag{6.1}$$

where S_T is the value of the portfolio at time T.
The annualized volatility for the same period T is given by

$$V_T = \sigma(S_T) * \sqrt{\frac{1}{T}}. \tag{6.2}$$

R_T and V_T are computed for the two strategies and compared, so that the performance of the models is evaluated in using both yield and risk.

Another way to determine the performance of a trading rule would be to calculate the Pessimistic Return Ratio (PRR). This is a profit factor measure adjusted to give more importance (bigger weight) to more reliable trading times. It is given by the following formula [24]:

$$PRR = \frac{(((W - \sqrt{W})/T) * AW)}{(((L + \sqrt{L})/T) * AL)} \tag{6.3}$$

where

- W is the number of winning trades;

- L is the number of losing trades;
- T is the total number of trades;
- AW is the average winning trade amount;
- AL is the average losing trades amount.

PRR measures greater than 2 are indicative of a good system and over 2.5 of an excellent system. Unlike other performance estimates, this method takes into account the number of effective trades, which is valuable since an approach that trades frequently is hit by larger trading costs.

6.1.2 Trading rules experiments

We combined the prediction model given by the cross-validation process with the simplest trading rule. The limits were fixed at 40% and 60%. Every time a trade was made, all the the capital was invested or all the assets were sold. Table 6.1 presents the summary of the results for an initial investment of 100. We will only present results for the strings of length 6, since the results for strings of length 12 were consistently worse.

Index	SL	SW	alpha	lambda	ar BS	ar BH	va BS	va BH	PRR
SX5E	60	6	0.08	1.8	14.14	15.39	8.0013	8.1966	1 purch
NDX	40	6	1	1	7.93	8.79	6.5518	6.7229	1 purch
RTY	6	6	0.01	2.8	14.05	6.46	6.5423	5.4452	2.1475
SPX	13	6	0.08	7.1	9.45	9.95	5.8234	5.879	1 purch
UKX	50	6	0.11	18	9.67	5.51	4.8569	3.612	2.5457

Table 6.1: performance of the trading strategies with the cross validation estimations where:
SL=*size of levels,* ***SW***=*size of windows,* ***arBS***=*annualized return for the binary string strategy,* ***ar BH***=*annualized return for Buy & Hold Strategy,* ***va BS***=*annualized volatility for the binary string strategy,* ***va BH***=*annualized volatility for Buy & Hold Strategy,****purch***= *number of purchases during the period.*

The performance is better than a random walk process for the RTY and UKX indices. For these series, the return and the volatility overperformed significantly the

B& H strategy and the PRR is better than 2 which is supposed to indicate a good trading rule. However, we must be careful with this measure because the strategy only sold stocks fewer than 40 times and only trade have a negative return. It seems to especially mean that most of the trades are successful and that trade are seldom compared to the test set length. The other experiments achieve worse performances than random walk. This could be explained by the fact that the trading rule made a single trade to buy the assets and keep them until the end. Unfortunately, it bought the stocks at a point when they more expensive than the Buy & Hold strategy (which buys them at the very start of the time period. In this case, the problem is due to a bad choice of the thresholds or of the parameters. In addition, these trading strategies were applied to the models obtained using maximum likelihood parameter estimation and the trading rule achieved worse results than the Buy and Hold Strategy whatever the thresholds. The following figures illustrate these two cases:

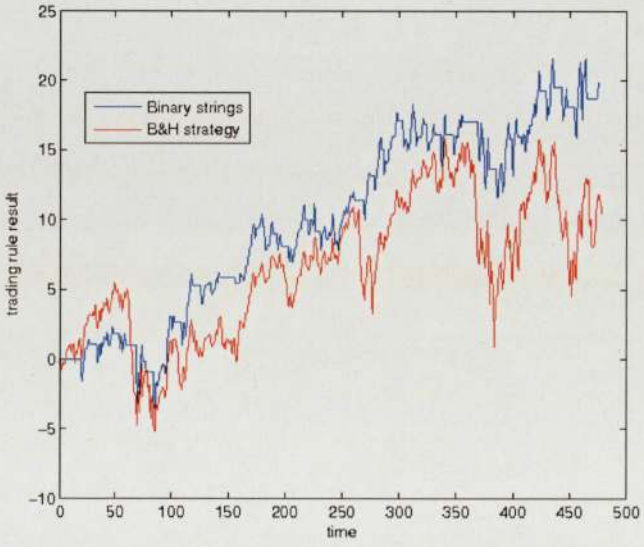

Figure 6.1: Trading rule for the RTY index for strings of size 6.

On this curve, the trading rule is particularly effective at avoiding losses. In order to overperform the index when the prices increase, it would be of interest to apply more complicated trading rules including short sales.

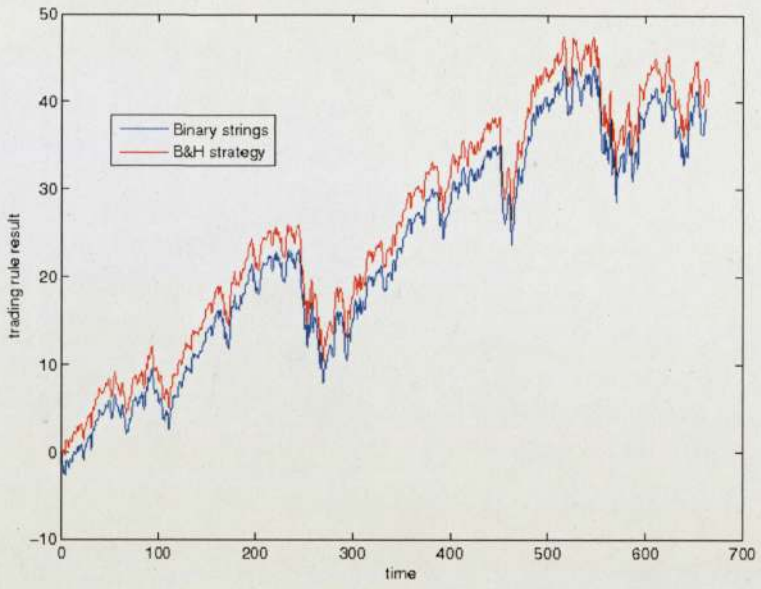

Figure 6.2: Trading rule for the SX5E index for strings of size 6.

As this graph is not very conclusive regarding the trading point of view, we computed the probabilities given by the model in order to modify the thresholds of the strategy:

The result of the trading strategy can be explained by the lack of small probabilities. The value of the low limit was changed to 0.48 and a best model was obtained with an annualized return of 18% , an annualized volatility of 9.6720 and a PRR greater than 5 which is illustrated in the following figure:

This model almost classifies the strings in two separated sets, one of strings followed by an "up" and the other one strings followed by a "down". It shows that the model can be improved by varying the thresholds of the trading strategy.

6.1.3 Relationship between the parameters of the model and of the trading strategy

From the experiments in the last chapter, we deduced that if we change the values of the parameters by a small amount, the model accuracy does not vary. The initial experiments reported in this chapter show that, by moving the thresholds of the decision

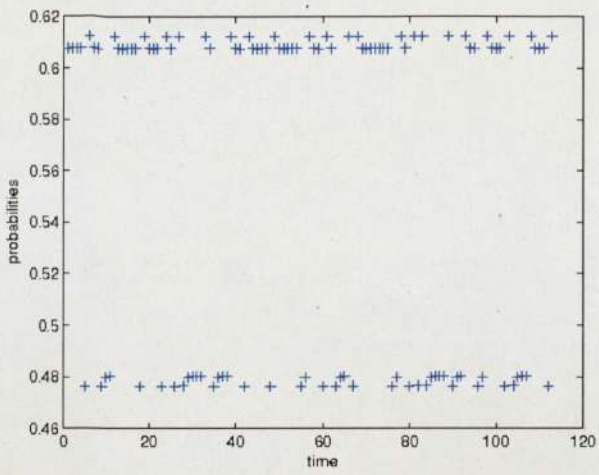

Figure 6.3: Probabilities for the SX5E index for strings of size 6.

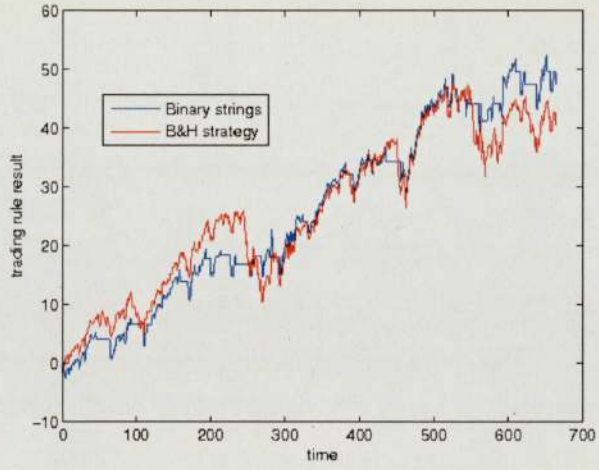

Figure 6.4: Trading rule for the SX5E index with a low limit of 0.48.

function of the computed probabilities, the performance of the trading rule can be improved. Following on from this, experiments have been run with samples of the pair (α, λ) and 2 pairs of thresholds $(40\%, 60\%)$, $(45\%, 55\%)$. Using these tests, we would like to find empirical evidence of a relationship between (α, λ) and the thresholds. We ran these experiments for all the indices and with training sets of 300 strings as advised by the analysis of string entropy: indeed, it is sufficient because larger sets have been tested empirically and this does not improve performance.

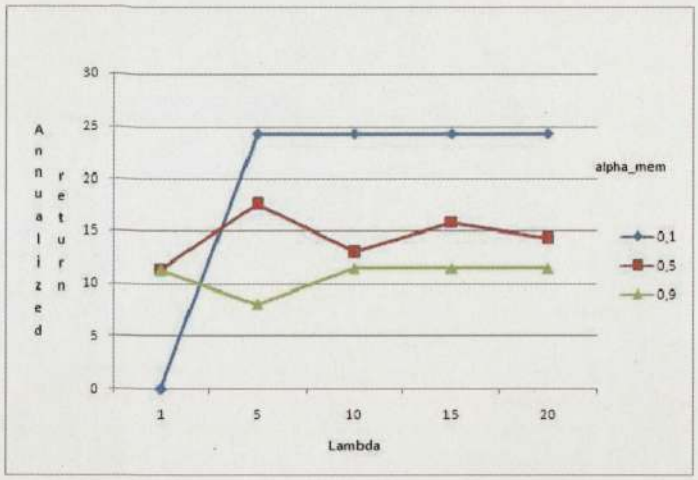

Figure 6.5: Annualized returns for the SX5E index (limits 40% and 60%).

Figure 6.5 represents the graphs of the annualized returns as functions of α and λ. The maximum is obtained for $\alpha = 0.1$ and for λ between 5 and 20. As was stated before, the model performance is steady over smaller parameter variations. For the other indices, the optimum values of (α, λ) change but the stability remains especially related to λ. We observe that all the indices achieve a better performance than the Buy & Hold strategy as the following table illustrates:

index	delta	sw	alpha	lambda	arBS	arBH	vaBS	vaBH	mup	stdp
SX5E	5%	6	0.1	5–20	24.28	12.47	12.087	6.9952	0.552	0.1373
NDX	5%	6	0.9	5–20	17.34	10.37	10.346	7.1881	0.5809	0.1967
RTY	5%	6	0.1	5–20	14.36	4.69	6.004	5.2475	0.5403	0.1649
SPX	5%	6	0.9	10	10.51	8.64	5.3621	5.5795	0.514	0.2335
UKX	5%	6	0.1	5–20	9.82	3.79	4.9543	3.5938	0.5348	0.1105

Table 6.2: Trading rule performance with the threshold $(0.4, 0.6)$.

Moreover the volatility curves show the same shapes as the returns. It confirms that the higher are the risks higher should be the returns which is a very well-known rule in finance.

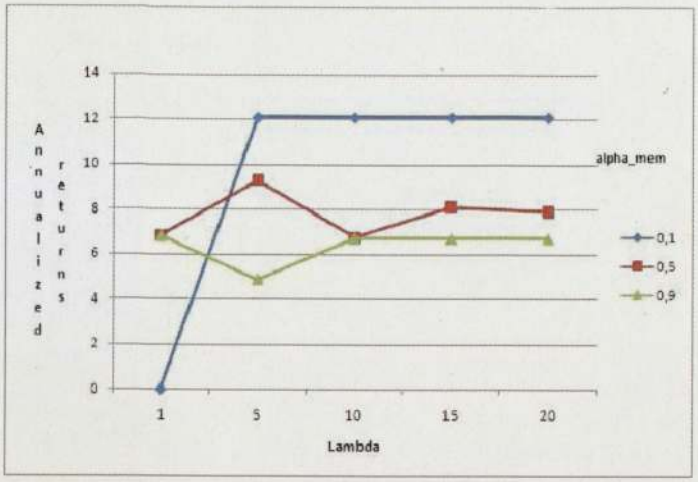

Figure 6.6: Annualized volatility for the SX5E index.

By using a narrower threshold $(45\%, 55\%)$ for the trading strategy, globally better returns were obtained whatever the parameters:

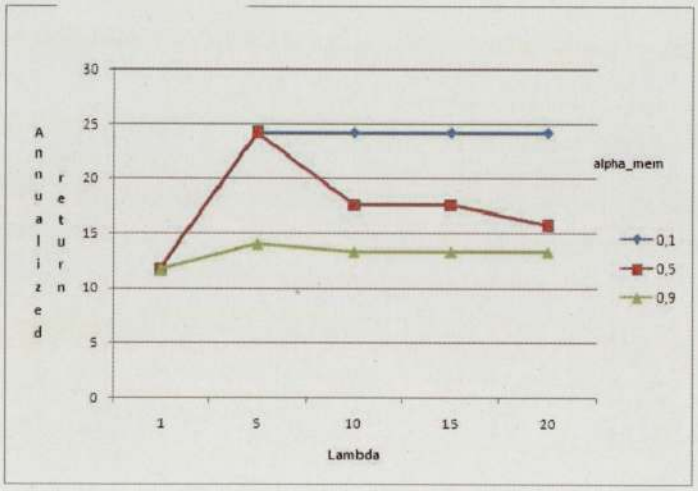

Figure 6.7: Annualized returns for the SX5E index (0.45% – 0.55%).

This table summarizes the information obtained with narrower limits:

index	δ	sw	alpha	lambda	arBS	arBH	vaBS	vaBH	mup	stdp
SX5E	5%	6	0.1;0.5	5–20;5	24.28	12.47	12.087	6.9952	0.552	0.1373
NDX	5%	6	0.9	5–20	19.48	10.37	10.57	7.1881	0.5809	0.1967
RTY	5%	6	0.1;0.5	5–20;5	14.36	4.69	6.004	5.2475	0.5403	0.1649
SPX	5%	6	0.9	5	12.14	8.64	6.7089	5.5795	0.5185	0.2003
UKX	5%	6	0.5	20	11.37	3.79	5.5521	3.5938	0.5381	0.1528

Table 6.3: Trading rule performance with the threshold $(0.45, 0.55)$. $(0.1; 0.5)$: *α=0.1 or 0.5,* $(5 - -20)$: *λ= 5 to 20.*

As the returns are greater for a narrower threshold, it seems that these limits are better suited to the set of predicted probabilities from the model. When the model outputs are almost separated in two classes, we should choose limits which permit as many trades as possible. But the distribution takes this shape only for small $\alpha \sim 0.1$ and large $\lambda > 5$ or small $\lambda \sim 1$. Moreover, some indices (e.g. SPX) reached their best performance with the biggest values of parameters. Then, when the parameters increase, the distribution becomes more and more continuous, as shown in Figure 6.8:

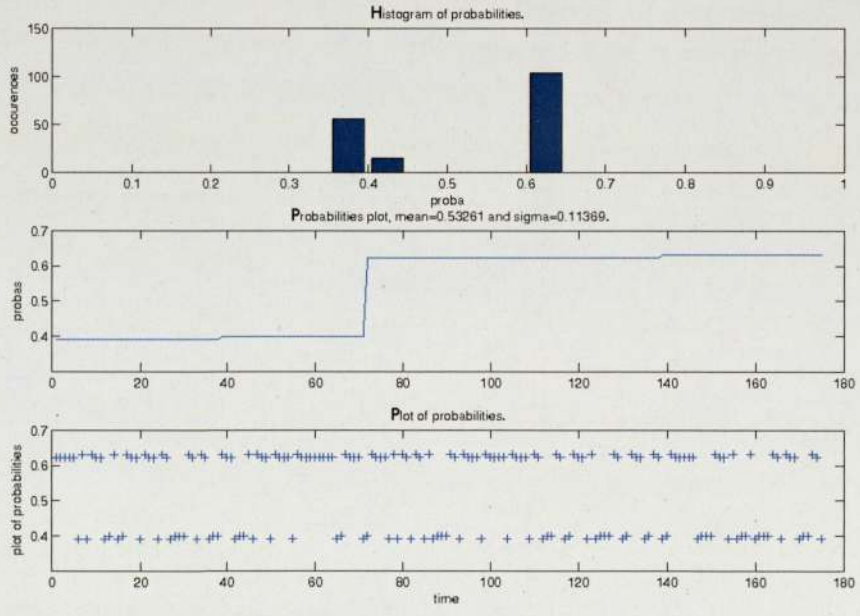

Figure 6.8: Histogram of probabilities distribution for the SPX index $\alpha = 0.1$ and $\lambda = 5$.

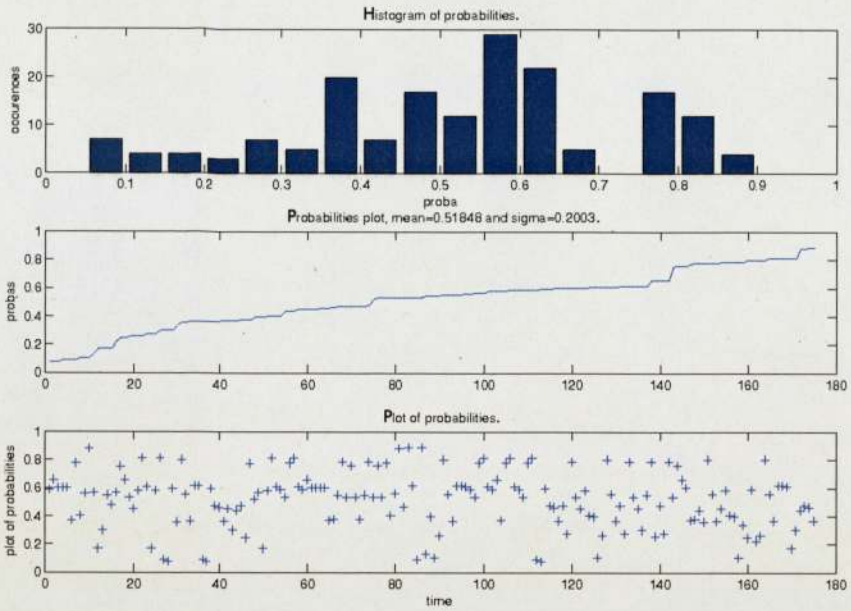

Figure 6.9: Histogram of probabilities distribution for the SPX index $\alpha = 0.9$ and $\lambda = 5$.

The conclusion of these experiments is that the performance of the model is strongly dependent on the relationship between the pair (α, λ) and the thresholds. To construct a profitable trading rule from on this task there are two solutions:

- From a small α or λ, the computed model gives a classification and then the limits are chosen in order to perform the most trade as possible.
- With a medium lambda and $\alpha > 0.1$, the computed model gives a continuous probability distribution including extreme values, such as 10% and 90% and then the trading limits are adapted.

One way to do the adaptation of the thresholds is cross-validation. After separating the data between the training set, the validation set and the trading set, the annualized return and the annualized volatility are computed on the validation data. When the optimum values of the parameters is found, the new model is tested on the test set.

6.2 Moving memory trading rule

The prediction model does not take time into account (apart from the creation of delay vectors) even though this notion seems important in the financial market field. The data are considered as training patterns so the model will give the same output probabilities whatever the string order. In particular, we have assumed that the data generator is stationary over the training, validation and test sets. This is unrealistic, given the results in Chapters 3 and 4.

Our approach to address this was to try a new process including online learning: the number of training strings is fixed but their set is moving with the time. In other words, the training set is never the same, it evolves with time. A moving memory means that the training set which moves when the next bit is predicted: the first pattern of the training set is deleted and the more recent bit is added to this set.

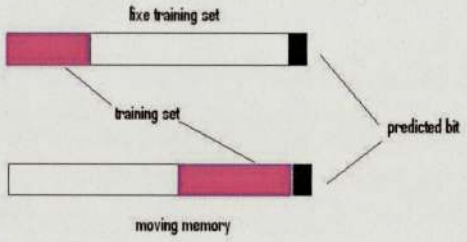

Figure 6.10: 1-prediction with a fixed training set.2-prediction with a moving training set

6.2.1 Kullback-Leibler Divergence

The idea of a moving string set assumes that the distribution of a fixed number of strings is stationary but that it changes over time (so that the series as a whole is non-stationary). From the histograms of the distributions, it is not possible to determine if the samples are close. An objective measure of distribution similarity is helpful to determine how large the moving string set should be. The Kullback-Leibler (KL) divergence is a fundamental equation of information theory that quantifies the proximity

of two probability distributions. For each index the array of sequences was divided into sets of same cardinality from 200 to 400 samples. We computed the KL divergence between sets of the same cardinality.

The Kullback-Leibler Divergence [26] between two probability distributions P and Q is given by the following equations:

P_i, probability that the i^{th} sequence occurs: $P_i = \frac{nb(i)}{nbtotal(sequences)}$.

$$D_{KL}(P||Q) = \sum P_i \log_2 \frac{P_i}{Q_i}. \tag{6.4}$$

Note that the KL divergence $D_{KL}(P||Q)$ is not symmetric in P and Q.

$$D_{KL}(Q||P) = \sum Q_i \log_2 \frac{Q_i}{P_i}. \tag{6.5}$$

We defined an overall KL distance as follows

$$D(P,Q) = \frac{1}{2} * (D_{KL}(P||Q) + D_{KL}(Q||P)). \tag{6.6}$$

The values obtained were less than 0.40. In order to evaluate the significance of these results and hence determine the optimum size of set, the method of surrogate sets was applied [15]: this process is convenient to estimate the validity of an hypothesis without using a distribution-based statistical test and is consequently a useful concept in system evaluation, particularly when working out the relevant distribution for the null hypothesis is difficult. It is a statistical test, but it is distribution-free and does not require any theoretical analysis of the distribution of a parameter under the null hypothesis. The idea is to generate surrogate data sets sharing characteristics of the original data (e.g. permutations of series have the same mean, variance etc.) and for each surrogate to compute a statistic of interest, here the KL divergence. If a large proportion of the surrogates give more extreme statistics than the original series, then the null hypothesis (that the result is not significant) can be rejected.

It is assumed that there are two initial sets S1 and S2 and we would like to know if their distributions are similar in order to understand if the moving sets are stationary. Then, the null hypothesis is stated:

H_0: the probability distributions generated by the pre-processing are stationary.

The surrogate data are generated by randomly constructing sets of smaller cardinality from S1 and S2 and the KL divergence is computed between all these subsets to determine the optimal length for the moving memory. The following plot was obtained for the SX5E index:

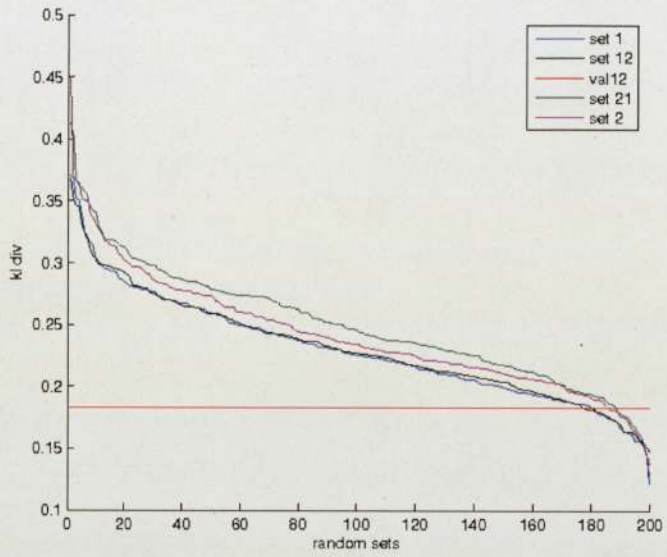

Figure 6.11: Method of surrogate sets: KL divergence.

On the graph, all the curves are grouped together and this was also found to be the case for all the indices and all size of sets. The percentage of points under the horizontal line (which represents the KL divergence on the original datasets) is calculated. We observed that the number of surrogate sets whose KL divergence is smaller than the KL divergence computed for the two initial sets decreases when the cardinality of the sets increases. It seems to suggest that the larger training sets are more stationary. But when we compared the results for the UKX index, we notice some inconsistent percentages, for the same cardinality, between different sets, the method of surrogate sets gave very varied values like 1.5%, 91% and 33.5%.

sl	sw	nbdata	nbset	KLdiv1/2	KLdiv1/3	KLdiv2/3	prct1/2	prct1/3	prct2/3
50	6	250	3	0.15	0.26965	0.21375	1.5	91	33.5
50	6	300	3	0.17	0.1826	0.28765	3.5	12.5	78.5
50	6	350	2	0.21			18.5		
50	6	380	2	0.21			14		
50	6	400	2	0.13			0		

Table 6.4: Method of surrogate sets where:
sl=*size of level,* ***sw***=*size of windows,****nbdata***=*number of data in the original sets,****nbsets***=*number of original sets,****KLj/k***=*KL divergence between the sets j and k,****PRj/k***=*percentage of KL divergence computed from the surrogate sets under the KL divergence calculated between the sets j and k.*

The results of these experiments are too variable to give a conclusion and we are going to test the performance of moving memory strategy within next chapter.

6.2.2 Performance of the moving memory rule.

Despite these results, the moving windows trading rule was tested on the indices. The only change compared with the original models was the use of the training set. Indeed for each prediction, the data set used to compute the distance is different because it moves with time. So the training set is defined by the 300 last windows before the predicted bit. The probabilities generated by the model were computed for this strategy:

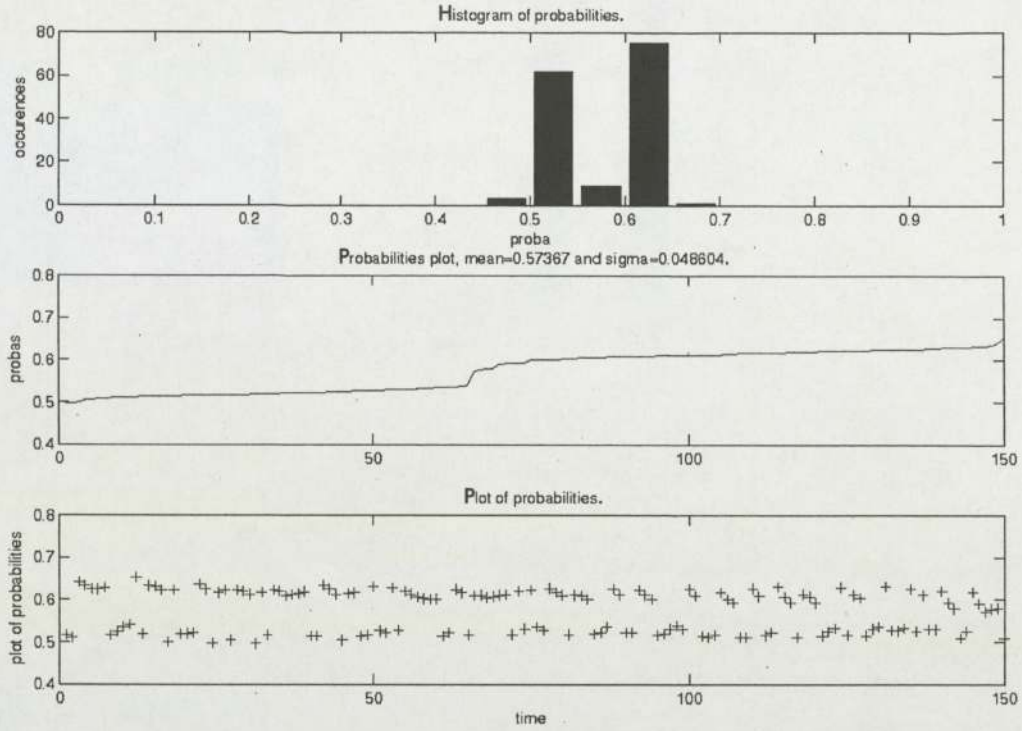

Figure 6.12: Histogram of probability distribution for the RTY index $\alpha = 0.5$ and $\lambda = 1$.

This distribution is similar to the ones obtained with the original model but the classification model is different. Indeed, with a small α, the probabilities took only four different values. Here, we observe almost a line but it is not straight. On the other hand, when α and λ become bigger, the distribution spreads out.

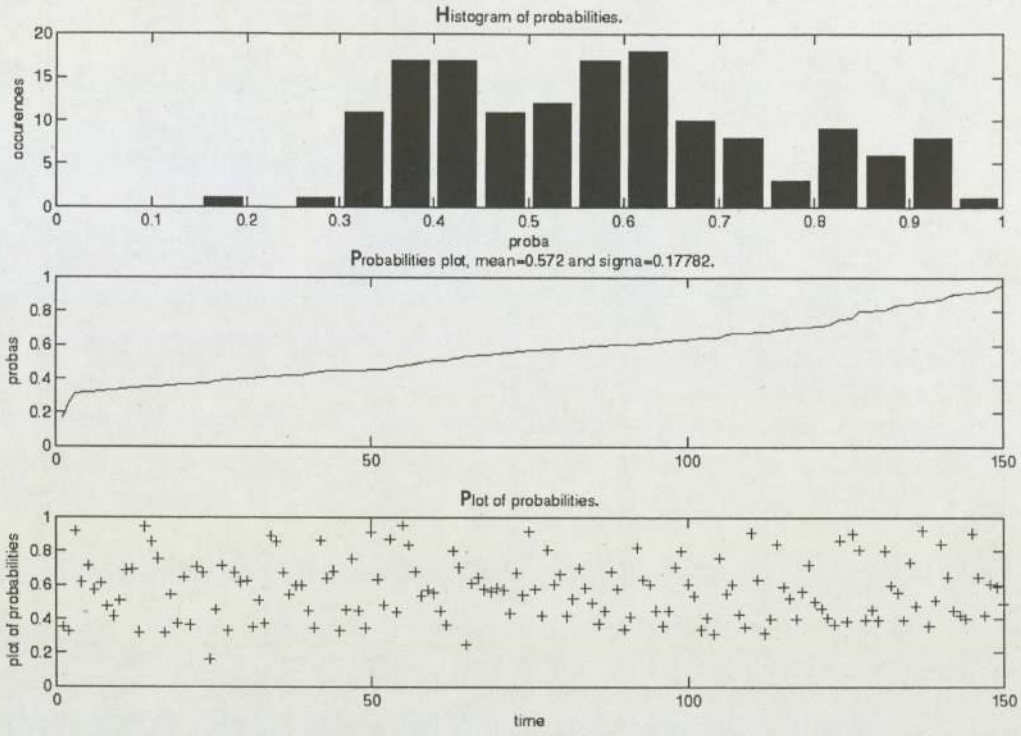

Figure 6.13: Histogram of probability distribution for RTY index $\alpha = 0.5$ and $\lambda = 20$.

If these predictions are compared to the ones obtained with a fixed training set (figure 6.8), one notice that the histogram is shifted right. So this model tends to predict more "up" than "down".

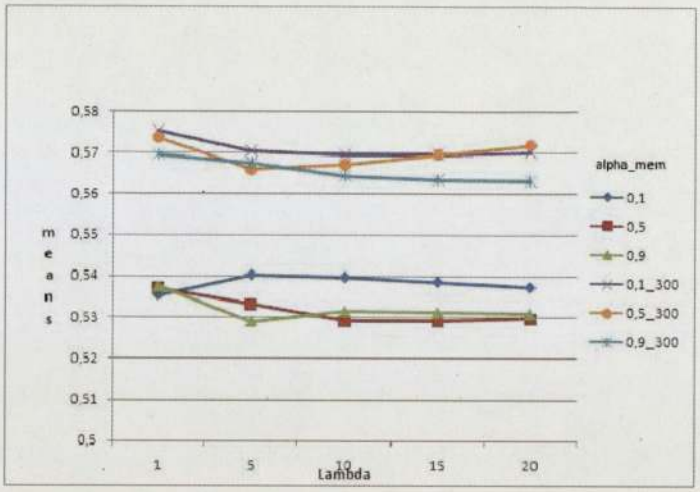

Figure 6.14: Evolution of the mean of the probability distribution.

The distributions obtained for the two trading rules are distinct as the mean evolution of the computed probabilities is higher for the moving memory model than for the fixed training set. The volatilities of the computes probabilities evolve also in two separate sets but, conversely, they are smaller with this new method. This explains the trend of this moving windows model to predict more "up" than the last one. As before the trading rule is computed and the annualized return and the annualized volatility are calculated.

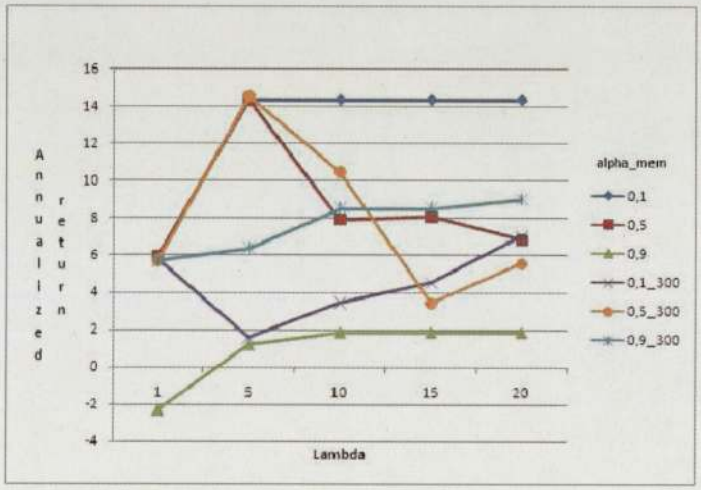

Figure 6.15: Annualized returns for the RTY index for the thresholds $(0.45, 0.55)$.

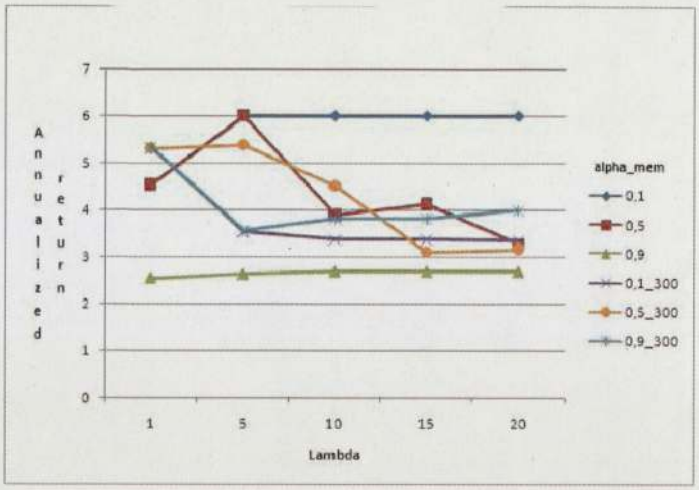

Figure 6.16: Annualized volatility for the RTY index for the thresholds $(0.45, 0.55)$.

Globally, the results obtained with the moving training sets achieve worse returns than the fixed training sets. But, for the RTY index, with $\alpha = 0.5$ and $\lambda = 5$, the new model achieves the best returns.

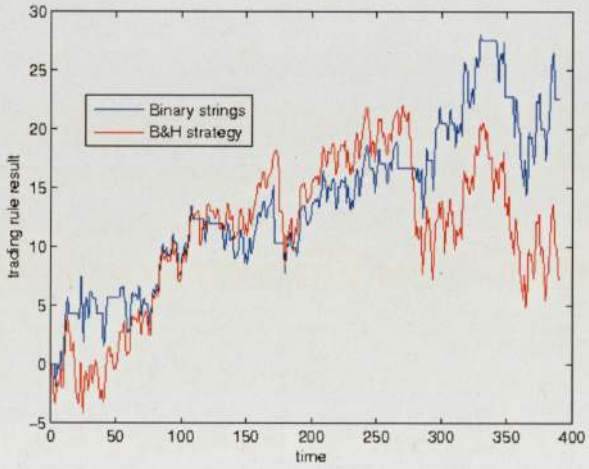

Figure 6.17: Trading results for the RTY index $\alpha = 0.5$ and $\lambda = 20$.

Nevertheless, for the other indices the trading performance is worse. To conclude, using a moving training set in the model does not necessarily achieve better results.

Chapter 7

Conclusion

The purpose of this thesis was to try a new approach in order to predict the behaviour of times series. Instead of forecasting the prices, we implemented Dupire's approach and predicted the next move that is to say if the market is going up or down. Therefore, we studied 5 financial time series which were 5 main indices of financial markets. These indices had the same main features as financial time series: non-stationary and fat-tailed distribution. We converted the times series into a string time series using Dupire's process. Afterwards, we studied the entropy of these new data and found out that the maximum was reached for strings of size 12. Furthermore we tried to classify the strings using logical regression and multilayer perceptron but the model tends to predict more up than down. Then we used cross validation and log-likelihood estimation to determine the values of the parameters of the model and implemented the model considering the parameters adapted to the data. We computed and compared the error generated by Dupire's model and a random process and concluded that the order of magnitude was almost the same. We then applied the model to a trading strategy to find out if it permitted to make profits. Most of the time, when the strategy ordered a trade, this was a winning trade. So in average, when the parameters are correctly determined, we make money and more money than a simple Buy and Hold Strategy. This was tested with a fix training set and a moving training set as it was explained within the last chapter. But most of the time a moving memory model does not achieve better performance. However, from our experiments we can list several modifications

which could improve the model.

- We observed that the trading rule avoids losing money, which is "down" prediction. Indeed, "short sales" is a convenient method to take advantage of a falling market. We could imagine a more sophisticated trading strategy which would say: "if the next bit prediction is a "1", borrow some money or buy options in order to purchase more assets and then return it after have sold everything." This process implies the access to supplementary data like option prices or interest rates.

- To really compute the exact returns and apply the process to stock prices , there are more parameters to take in account like compounding, dividend payments, and slippage (price movements between making a decision and executing a trade).

- We considered daily values, so that implied a limited set of data. Therefore, the number of trades is also restricted. We expect that by training models on higher frequency data, the results will be more significant and probably more conclusive.

At the end of this report, we tried to include take account of non-stationarity in the training set by using moving training sets. The experiments proved that for most indices the returns reduced. Nevertheless, there are some other ways to keep this idea. It should be possible to find a decreasing function for λ which will give more weight to closer windows than distant ones. Furthermore, it is important to take particular care in choosing the training sets. If we consider the totality of the price history, the model will be too long to compute. If we chose the training data too far-off in time from the prediction, it may be too different than the current situation to achieve good forecasts. If the string distribution is too different from the test sets, that will also reduce performance.

To conclude, the empirical results we obtained are encouraging but several potentially interesting tracks remain that we didn't explore because of a lack of data or a lack of time.

;

Bibliography

[1] Cottrell A. and Lucchetti R. *Gretl guide, Gnu regression, Econometrics and Time Series*, July 2008.

[2] Darling D. A. Asymptotic theory of certain goodness of fit criteria based on stochastic processes. *Annals of Mathematical Statistics*, 23:193–212, 1952.

[3] Dickey D. A. and Fuller W. A. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, 74:427–431, 1979.

[4] Foster A. Commodity futures price prediction: an artificial intelligence approach. Master's thesis, The University of Georgia, 2002.

[5] Lendasse A. Fast bootstrap applied to ls-svm for long-term prediction of time series. In *In proceding of the 2004 IEEE International Joint Confrerence on Neural Networks*, pages 705–710.

[6] Dupire B. Optimal process approximation: Application to delta hedging and technical analysis. Cambridge conference, July 2005.

[7] Mandelbrot B. *Fractals and Scaling in Finance.* Springer, 1997.

[8] Thomas J. D. *News and Trading Rules.* PhD thesis, Carnegie Mellon University, 2003.

[9] Said E. and Dickey D. A. Testing for unit roots in autoregressive moving average models of unknown order. *Biometrika*, 71:559–607, 1984.

[10] Shannon C. E. Prediction and entropy of printed english. *Bell Systems Technical Journal*, 27:379–423, 1951.

[11] Fama F. Random walks in stock markets prices. *Financial Analysts Journal*, 21:55–59, 1965.

[12] David J. H. and Saul D. J. *Statistics in Finance.* Arnold, 1998.

[13] Kutsurelis J. *Forecasting financial markets using neural networks: An analysis of methods and accuracy.* PhD thesis, Naval Postgraduate School, 1997.

[14] Murphy J. *Technical analysis of the financial markets: A comprehensive guide to trading methods and applications.* Prentice Hall, 1997.

[15] Theiler J., Eubank S., Longtin A., Galdrikian B., and Farmer J. D. Testing for non linearity in time series: the method of surrogate data. *Physica*, 58:77–94, 1992.

[16] Muller K-R., Smola A., Rtsch G., Scholkopt B., Kohlmorgen J., and Vapnik V. *Using support vector machines for time series prediction.* MIT Press, 1997.

[17] Molgedey L. and Ebeling W. Entropy and predictability of financial time series. *The European Physical Journal*, 15:733–737, 2000.

[18] Bishop C. M. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[19] Crisan M. About various definitions of entropy and their implications in cognitive sciences,. *Sci. & Tech. Bulletin of University "Politehnica" of Timisoara*, 40(54), 1995.

[20] Hutter M. Sequential prediction based on algorithm complexity. *Journal of Computer and System Sciences*, pages 95–117, 2006.

[21] Jensen M.C. Random walks and technical theories: Some additional evidence. *The Journal of Finance*, 25:469–482, 1969.

[22] Kohavi R. A study of cross-validation and bootstrap for accuracy estimation and model selection. pages 1137–1143. Morgan Kaufmann, 1995.

[23] Steuer R., Molgedey L., Ebeling W., and Jimenez M. Entropy and optimal partition for data analysis. *The European Physical Journal*, 19:265–269, 2001.

[24] Vince R. *Portfolio Management formulas.* Wiley Finance Edition, 1990.

[25] Dablemont S. and Simon G. Time series forecasting with som and local non linear models-application to the dax30 index prediction. In *Proceedings of the workshop on self-organizing maps, Kitakyushu, Japan*, pages 340–345, 2003.

[26] Kullback S. and Leibler R. A. On information on sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

[27] Tsay S. *Analysis of Financial Time Series.* Wiley Series in Probabilities and Statistics, 2002.

[28] Zemke S. *Data Mining for Prediction Financial Series Case.* PhD thesis, The Royal Institute of Technology, 2003.

[29] ROSS S.M. *An elementary introduction to mathematical finance.* Cambridge University Press, 2003.

[30] Mills T. *Time Series Techniques for Economists.* Cambridge University Press, 1990.

[31] Mitchell T. *Machine Learning.* McGraw Hill, 1997.

[32] Nabney I. T. *Netlab: Algorithms for Pattern Recognition.* Springer, 2002.

[33] Adampopoulos A. V., Likothanassis S., Pavlidis N. G., and Vrahatis M. N. Icnaam-extended abstract,1-4, short-term prediction of complex binary data, 2005.

[34] Brock W., Lakonishok J., and Lebaron B. Simple technical trading rules and the stochastic properties of stock returns. *The Journal of Finance*, 47:1731–1764, 1992.

[35] Ebeling W., Molgedey L., and Kurths J. *Complexity, Predictability and Data Analysis of Time Series and Letter Sequences.* Springer, 2002.

[36] Campbell J. Y., Lo A. W., and MacKinlay A. C. *The econometrics of financial markets.* Princeton Univesity Press, 1996.