T H E N A T I O N A L U N I V E R S I T Y
of S I N G A P O R E

S c h o o l o f C o m p u t i n g
Lower Kent Ridge Road, Singapore 119260

TRD1/05

A Comparison between MDPP and Kernel Regression smoothing techniques for Forecasting Time Series Data

Deatrice LUCA and Siau-Cheng KHOO

January 2005

TechnicalReport

Foreword

This technical report contains a research paper, development or tutorial article, which has been submitted for publication in a journal or for consideration by the commissioning organization. The report represents the ideas of its author, and should not be taken as the official views of the School or the University. Any discussion of the content of the report should be sent to the author, at the address shown on the cover.

JAFFAR, Joxan
Dean of School

A Comparison between MDPP and Kernel Regression smoothing techniques for Forecasting Time Series Data

Beatrice Luca and Siau-Cheng Khoo

Department of Computer Science
National University of Singapore

January 20, 2005

Abstract

We present a comparison between two smoothing techniques for forecasting time series, Minimal Distance Percentage Principle(MDPP) and non-parametric Kernel Regression. This comparison reveals some important aspects such as the choice of the smoothing parameters, the complexity of the algorithms and the accuracy of the smoothing results used in forecasting. We have chosen to make tests on the predictive power of "head-and-shoulder" pattern .

Financial analysts are overloaded with huge historical stock data which makes searching for chart patterns a difficult job. This is the reason why we need to take smoothing as an integral part of the pattern definition, i.e to find, through the smoothing process, those landmark which are important in the pattern formation. Our final goal is to find as many good instances of "head-and-shoulder" pattern using MDPP and Kernel Regression smoothing techniques, and to forecast the future time series prices.

1 Introduction

It is an established notion among financial analysts that stock prices move in patterns and that these patterns can be used to forecast future prices. There have been many attempts among both researchers and practitioners to forecast prices more accurately, and for centuries, traders have relied on something other than numbers: pictures of price movement or price chart. In price charts, patterns reoccur and every time they bode of similar future events, the eternal struggle between "bulls" and "bears". For example, the famous "head-and-shoulder" predicts a substantial decline of price. Therefore, the goal of an investor is to discover meaningful patterns that are able to forecast the price movement from historical market data.

Technical analysis has not received the same level of academic scrutiny and acceptance as traditional approaches, such as fundamental analysis. One of the main obstacles is the highly subjective nature of technical analysis: the presence of geometric shapes in the historical price is often in the eyes of the beholder. Therefore, it is difficult to dispute the potential value of price movement, when confronted with the visual evidence. This idea underscores an important

difference between technical analysis, which is primarily visual, and quantitative finance, which is primarily algebraic and numerical.

The size of the historical data which the analysts want to search is huge and it is too expensive to work with raw data, even for the simplest similarity problem. Furthermore, in the real world, noise comes with almost every measurement. Conventionally, noise is considered obstructive to accurate forecasting, that is why there is a need to smooth the past data before looking for significant patterns. To achieve accurate forecasting, it is important to quantify the effect of noise on time-series prediction. On the other hand, studies were conducted to evaluate the performance of the forecasting on noise-injected time-series and the results are highly encouraging [18].

In this paper, we compare two smoothing techniques, Minimal Distance Percentage Principle and non-parametric Kernel regression. The former is a smoothing technique based on a "landmark model", and the latter is one of the most common methods for estimating non-linear relations. The aim of this comparison is to reveal important aspects which can have great impact on the accuracy of forecasting results, such as the choice of the smoothing parameters, the efficiency of the algorithm, etc.

Our final goal is to identify regularities in the time series of prices by extracting non linear patterns from noisy data, i.e. to find as many "good" pattern instances as possible. "Good" does not always mean optimal, but rather effective. As Andrew W. Lo said, "patterns that are optimal for detecting statistical anomalies need not be optimal for trading profits, and vice versa" [15]. Having this objective established, we seek to find important characteristics of efficient smoothing technique applied to time series and to treat smoothing process as an integral part of pattern definition process.

2 Related work

In the area of compressing time series we found particularly interesting the work of Perng et al. [5] who investigated a compression procedure based on extraction of "landmark points" which include minima and maxima. In the area of research into data compression, Keogh and Pazzani [13] used the endpoint of best-fit line segments to compress a series. Keogh et al. [12] reviewed and compared the compression techniques based on approximation of a series by a sequence of line segments.

There are some important studies conducted to assess the effectiveness of Kernel Regression in detecting technical patterns [15, 9]. In [15], kernel regression is applied to a large number of US stocks, from 1962 to 1966, to identify a variety of technical patterns and to find statistical evidence that there is potentialy useful information included in the tested patterns. They suggest that technical analysis can be improved by using automated algorithms that do not look for "optimal" patterns but rather for patterns that optimize specific objective functions. Savin [9] uses a modified approach based on Andrew Lo paper [15] in order to evaluate the predictive power of the "head-and-shoulder" pattern in Russell 200 stocks and Standard & Poor's 500 (S & P 500) stocks, over a period between 1990 to 1999. They obtained strong evidence that the "head-and-shoulder" pattern has the power to predict excess returns.

We have developed a new technique for compressing time series data based on landmark model and MDPP [5]. The primary effort in this paper is the comparison between MDPP and Kernel

Regression smoothing techniques for forecasting time series data.

The main contributions of this paper include:

1. A revised algorithm for MDPP which extends the reach of Technical Analysis by taking the smoothing process as an intergal part of pattern definition. From our knowledge, no study has been conducted to evaluate the power of MDPP as a smoothing algorithm applied to the forecasting of time series data. We believe that a good smoothing technique for forecasting purpose should keep those "eminent" peaks and bottoms of the time series, given their tremendous importance in trading.
2. A comparison between two smoothing techniques, MDPP and non-parametric Kernel regression. The first one is a relatively new smoothing technique. The latter is one of the most common methods for estimating non-linear relations.
3. A general framework for forecasting time series that brings together smoothing techniques, pattern design/querying and forecasting methods for time series.
4. A comprehensive experimental evaluation based on data collected from 362 Standard & Poor ($S\&P$) 500 stocks over the period 1990-1999, using our Chart Pattern Language (CPL).

3 Forecasting Time Series

"`I have seen the future and it is very much like the present, only longer.`"

`Kehlog Albran,` *The Profit*

Good forecasting methods are vital in many areas of scientific, industrial, commercial and economic activity. Moreover, good forecasting techniques require integration into the management process as well as good computation. We are concerned only with *time series forecasting* where forecasts are made on the basis of past values. Past values, often referred to as a "time series", are spaced equally over time and can represent anything, from monthly sales data to daily electricity consumption, to hourly call volumes.

Time series forecasting assumes that a time series is a combination of a pattern and some random error and the goal is to separate the pattern from the error by understanding the pattern's *trend*, its irregular repetition (*cycling*) and its *seasonality* [3, 19]. We have to highlight the importance of knowing the features of a time series because methods which are good in one environment can be bad in another environment. For example, forecast methods that are good at detecting trends in stock prices often miss turning points between up trends and down trends. So, it is now well established that no single method will outperform all other methods, in all situations, and, in any case, it depends on what is meant by "best". Another assumption that we make is that the future will behave like the past and, given this assumption, there is a limit to how accurately a forecast can be. The achievable accuracy depends on the magnitude of the noise component.

Forecasting methods can be broadly classified into three types [17]:

1. *Judgemental forecasts* based on subjective judgement, intuition, "inside" commercial knowledge, and any other relevant information.

2. *Univariate methods* where forecasts depend only on present and past values of the single series being forecasted, possibly augumented by a function over time such as a linear trend.

3. *Multivariate methods* where forecasts of a given variable depend, at least partly, on values of the one or more additional time series variables, called predictor or explanatory variables. Multivariate forecasts may depend on a multivariate model involving more than one equation if the variables are jointly dependent.

A more detailed classification of forecasting methods is given in the Figure 3a below, which splits the forecasting methods into statistical and non statistical methods. The statistical methods can be univariate or multivariate. The non-statistical methods are based on technical analysis techniques and judgmental methods.

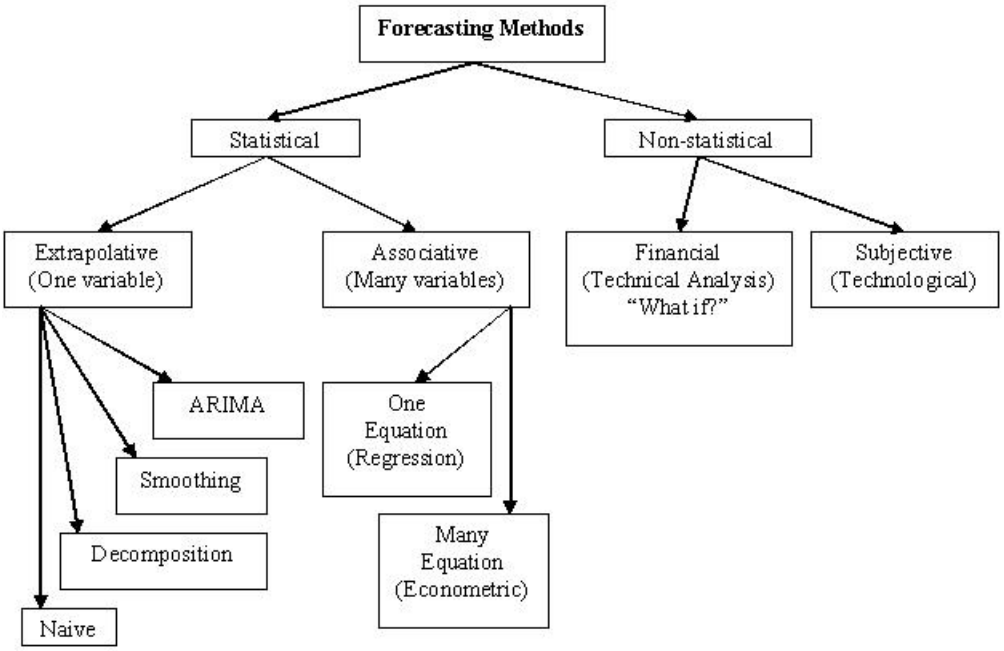

Figure 3a Clasification of forecasting techniques

More generally a forecasting method could combine more than one of the above approaches, as for example, when univariate or multivariate forecasts are adjusted subjectively to take account of external information which is difficult to express formally, in a mathematical model.

Armstrong and Collopy address in [17] the issue of integrating statistical methods and judgment for time series forecasting based on empirical research from 47 studies, almost all published since 1985. Their work draws conclusions from those studies in an effort to unite findings from different studies conducted under varying conditions. Integration is shown to improve accuracy when experts have good domain knowledge and when significant trends are present. Armostrong and Collopy indentify three conditions under which integration should be considered. First, one needs

quantitative data that have some relevance for the future. Second, judgemental inputs should provide additional relevant information beyond that contained in the statistical model and vice versa. Third, judgement should be unbiased.

Integration is most effective when judgement is used as an input rather than to revise the statistical output. A good starting point is to use equal weighting of statistical and judgemental forecasts as a benchmark, particularly when there is high uncertainty or instability in the series.

The choice of integration approach was found to have a substantial impact on forecast accuracy. When there is high instability in the historical data, the authors recommend revising judgment and revising extrapolations. Rule-based forecasting is recommended when good domain knowledge is available, when significant trends are present, and when there is low uncertainty. When future conditions are expected to contain trends that are contrary to expectations, econometric models are recommended.

Let us now answer an important question: "What is the purpose of forecasting?" The ultimate goal of forecasting is to represent knowledge as *forecasting principles*, to determine what actions to be taken in a given situation. Some examples of well known simple forecasting principles, not necessary optimal, are "Buy low and sell high" or "Collect early and pay later". Having this purpose, we need to find the proper instrument to develop a theory of economic behavior. The difficulties are in the proper application of the mathematical method. Usually, economic problems are not formulated clearly and are often stated in such vague terms. This makes mathematical treatment appears futile because it is quite uncertain what the problems are. That is why, economists must try to construct theories which explain very simplest facts, as well as, very hardest facts in economic life and which are conforming to rigorous scientific standards. The theory and application must corroborate with each other and beyond this lies the real success: *prediction by theory.*

To expedite the progress in developing method of good forecasting, researchers should directly study forecasting principles so that the practitioners, who seem to rely primarily on software rather then useful knowledge, can apply them in practice. In forecasting as in other areas of management science, we need to conduct research on principles, do meta-analysis to summarize knowledge as principles and to make principles easily available through papers, books, web sites and software.

Applications of forecasting include:

- inventory control/production planning - forecasting the demand for a product enables us to control the stock of raw materials and finished goods, plan the production schedule, etc;
- investment policy - forecasting financial information such as interest rates, exchange rates, share prices, the price of gold, etc;
- economic policy - forecasting economic information such as the growth in the economy, unemployment, the inflation rate, etc is vital both to governments and businesses in planning for the future.

It is advisable to include here a brief warning on the danger of forecasting. Time-series forecasting is essentially a form of extrapolation in that it involves fitting a model to a set of data and then using that model outside the range of data to which it has been fitted. However, when forecasting the future of a time series, extrapolation is unavoidable. A trader should always keep

in mind that the forecasts generally depend on the future being like the past and he should always be prepared to modify the forecasts in the light of any additional information, or to produce a range of different forecasts (rather than just one forecast) each of which is based on a known set of clearly stated assumptions. Even the long-term forecasting studies which are based on clearly stated assumptions and which are well worth doing, are still presumptious. Celebrated examples, which have not been forgotten, include the founder of IBM predicting "a world market for about five computers" in 1947, and the President of Digital Equipment predicting that "there is no reason for any individual to have a computer in their home" in 1977. Going further back into history, the U.S President Hayes in 1876, after he had witnessed a telephone call, said that it was "an amazing invention but who would ever want to use one?"

Of course, forecasts can even go horribly wrong in short-term when there is a sudden change or "structural break" in the data. One famous example is that made by Professor of Economics at Yale University 1929, when he said that "Stock prices have reached what looks like a permanently high plateau." This was just before the stock market "crash", which led on to the Depression!

4 Minimal Distance Percentage Principle (MDPP)

In this section we are concerned with the important characteristics of an efficient smoothing technique when applied to time series. We first describe the existing MDPP smoothing method and discuss its limitations. Next, we propose a new algorithm for smoothing time series data. In essence, we believe that a good smoothing algorithm should keep the "eminent" peaks and bottoms of the time series, given their tremendous importance in trading. In practice, all traders will choose to trade not at local maxima and minima but at major maxima and minima points. The desired smoothing algorithm will have to keep all *major maxima* and *minima* and smooth away all the points in between.

Based on the *Landmark Model*, which uses landmarks instead of raw data for processing, *Minimal Distance Percentage Principle (MDPP)* is a smoothing algorithm which runs in linear time. The landmark concept is relatively new and simple and was introduced by Perng in [5]. As the name suggests, the Landmark Model captures the relevant markers and reconstructs the time series by interpolating the missing data.

It will be helpful at this point to give some formal definitions.

4.1 Formal Definitions

Definition [Time Series] A *Time series (TS)* of length n is a sequence of time stamped data entries, ordered chronologically: $\{(t_1, y_1),...,(t_i, y_i),...,(t_n, y_n)\}$, allowing a natural association of data collected over intervals of time.

Our test data are obtained from daily summaries of real stock market, including open, close, high, low prices as well as volume data which are naturally modeled with regular time series. In this kind of time series, data arrive predictably, at predefined intervals.

Let us define further what local maxima and minima points are.

Definition [**Local Minima and Maxima points**] Given a time series TS = $\{(t_1, y_1),...,(t_n, y_n)\}$, (t_i, y_i) is considered a *local maxima* if and only if $(y_{i-1} < y_i)$ and $(y_i > y_{i+1})$. Similarly, a point (t_i, y_i) is a *local minima* if and only if $(y_{i-1} > y_i)$ and $(y_i < y_{i+1})$.

In [5], local minima and maxima points of a curve are called *first order points* because they are the zeros of the first order derivativ. We will adopt in this paper both names.

Assumption We make the assumption that the first and the last points from the time series are first order landmarks.

This assumption is necessary since we don't want to lose the borders of a time series after the smoothing operation.

Let us give now a first version of the definitions of major maxima and minima points. These definitions will be further improved.

Given a sequence of first order landmarks, FO = $\{(t_1, y_1),...,(t_m, y_m)\}$, obtained from a time series TS, we seek to find all those points which are major maxima and minima. The following property is trivially true.

Proposition 1 *Any major maxima (minima) is also a local maxima (minima); however, a local maxima (minima) need not be a major maxima (minima).*

Definition [**Major maxima**] A point (t_k, y_k), from FO is a *major maxima*, with respect to an interval $[i, j]$, where i $\leq$ k $\leq$ j, if and only if y_k is the maximum among all y's within this interval.

Definition [**Major minima**] A point (t_k, y_k), from FO is a *major minima*, with respect to an interval $[i, j]$, where i $\leq$ k $\leq$ j, if and only if y_k is the minimum among all y's within this interval.

We will describe further the algorithm introduced by Perng in [5] and show that it does not keep all major maxima and minima landmarks.

4.2 The MDPP algorithm proposed by Perng

Given a time series, $\{(t_1, y_1),...,(t_n, y_n)\}$, where y_i is some value at time t_i, the main algorithm takes as input two smoothing parameters, distance (D) and percentage (P), and outputs the smoothed time series data.

The algorithm, as described by Perng, consists of two steps. The first step filters the raw data of the time series to obtain a series of first order landmarks. Given the first order landmarks obtained in the first step, the second step will smooth out further using the following two smoothing conditions:

- $d < D$, where $d = x_{i+1} - x_i$ (1)
- $p < P$, where $p = \frac{2|y_{i+1}-y_i|}{|y_i|+|y_{i+1}|}$ (2)

The reason for choosing specific smoothing parameters is very intuitive. For example, if a stock trader trades once a week (5 business days) and regards a 5% gain or loss as significant, then he simply uses MDPP(5,5%) to smooth the data. This approach ensures that no price movement larger than 5% is smoothed out.

More specifically, the algorithm will remove both points, (t_i, y_i) and (t_{i+1}, y_{i+1}), if both conditions (1) and (2) are true. Because the MDPP algorithm consists of checking and eliminating of two landmarks at a time, we will call it, **Two-landmark** approach. This algorithm has the property of preserving the time series with less information and less error than other techniques, such as one using Discrete Fourier Transformation [5]. Now, we argue that Perng's algorithm does not preserve all the major points from the time series.

Proposition 2 *The "Two-landmark" MDPP smoothing algorithm does not preserve all the major minima and maxima points.*

To prove that Proposition 2 is true, we give a counter-example on five first-order landmark points from Figure 4.2a and apply Perng's MDPP algorithm, using $D = 6$ and assuming that p $\geq$ P between landmarks E and F and $p < P$ for the rest of the landmarks.

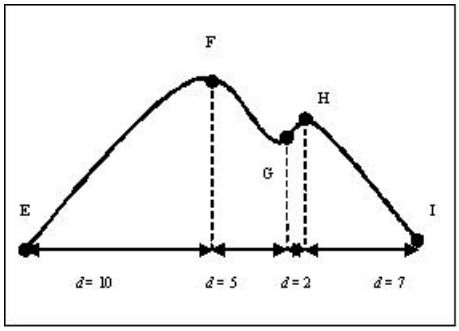

Figure 4.2a. Example of 5 first order landmarks.

Since the distance (E, F) is greater than 10 and the percentage is greater than P, we can conclude that E and F are significant, hence they are kept as mdpp landmarks. Then we proceed to check F and G points. As the distance (F, G) is less than 6 and p is less than P, F and G are considered insignificant, they are thus removed. Proceeding further, H and I points are not removed because distance (H, I) is greater than 6, even though p is smaller than P. Finally, we are left with landmarks E, H and I. This is illustrated by the dashed curve in Figure 4.2b. It is obvious that the dotted curve, with landmarks E, F and I would be a more appropriate smooth fit for the original curve. F is an important peak, a major maxima, that should not be removed since the d and p values between E and F are bigger than D and P respectively.

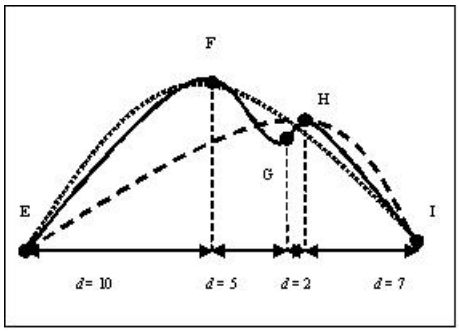

Figure 3.1b. The effect of "Two-landmark" MDPP algorithm

The dashed smoothed curve from Figure 4.2b misses the major maximum, F point. A refined curve is drawn, the dotted curve, corresponding to our corrected version of MDPP algorithm.

We conclude that the MDPP algorithm as described by Perng in [5] does not keep all the major maxima (minima) points, though it keeps all the first order points at the first stage. In Section 4.3, we will modify Perng's idea of smoothing by finding important landmarks.

4.3 A revised algorithm for extracting major minima (maxima)

As showed in the previous chapters, there is a need to smooth the past data before looking for significant patterns. In Perng paper, a preliminary step is to identify all the first-order landmarks from the raw data which are basically the local maxima and minima of the closing price time series.

For our investigation on 362 $S\&P$ 500 stocks , over a period of 10 years, approximately half of the raw data consists of first-order landmarks. Table 4.3(t1) shows a random example of 5 time series data with the number of the first order landmarks. For example, Microsoft (MSFT) contains 2528 raw data and the number of the first order landmarks is 1231 which represents 48.7% from the original points.

Company	IBM	MSFT	AAPL	MMM	BCC
Original Data	2528	2528	2528	2528	2528
First Order Landmarks	1227/48.54%	1231/48.7%	1190/47.07%	1216/48.10%	1130/44.7%

Table 4.3(t1) Originial data, the number of remaining landmarks / procent of the remaining landmarks generated by FindFo() algorithm.

The algorithm for computing first order landmarks, as described in Figure 4.3a, checks every point in the time series if it is a local minimum (maximum) and if this is the case, adds the point to the first order landmarks. It is possible, after one run of the algorithm, to have some plateau of points which are neither local maxima nor local minima. To be sure that every point is a first order landmark, we elect to further iterate the algorithm, as defined in Figure 4.3a.
Notations:
- p_i is the point at time i.
- close(p_i) is the close price at time i.

```
findFo() {
i=1;
addFo (p_i);
while (i < n)
                if ((close(p_{i-1}) < close(p_i) and close(p_i) < close(p_{i+1}))        /*check for non FO landmarks*/
                or (close(p_{i-1}) > close(p_i) and close(p_i) > close(p_{i+1})))
                or (close(p_{i-1}) == close(p_i) and close(p_i) == close(p_{i+1})))
                 deletenonFo (p_i);
                else i++;
}
```

Figure 4.3a Algorithm for computing first-order landmarks: begin with a copy of the raw data and delete the points which are not FO landmarks; the first and the last points are FO landmarks by default.

Figures 4.3b and 4.3c show the example of an IBM time series with raw data and first order landmarks.

Comparing the raw data and the first order landmarks, we observe that the effect of the algorithm for finding the first order landmarks is not significant and the data are still "noisy". Therefore, we need to further filter out the noise by applying the MDPP algorithm.

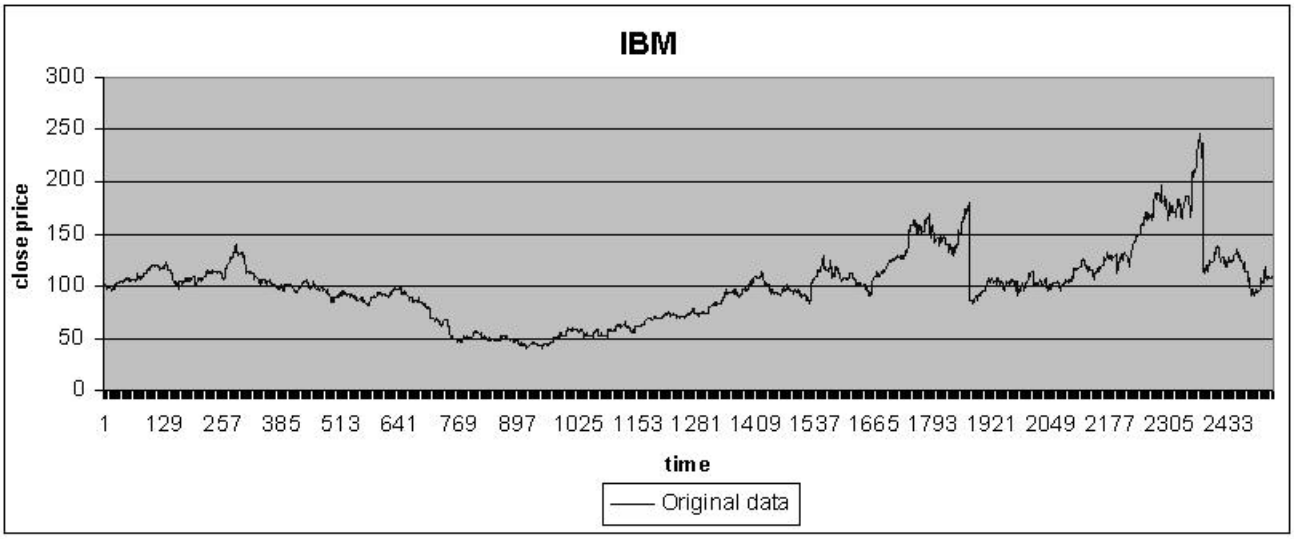

Figure 4.3b - IBM raw data.

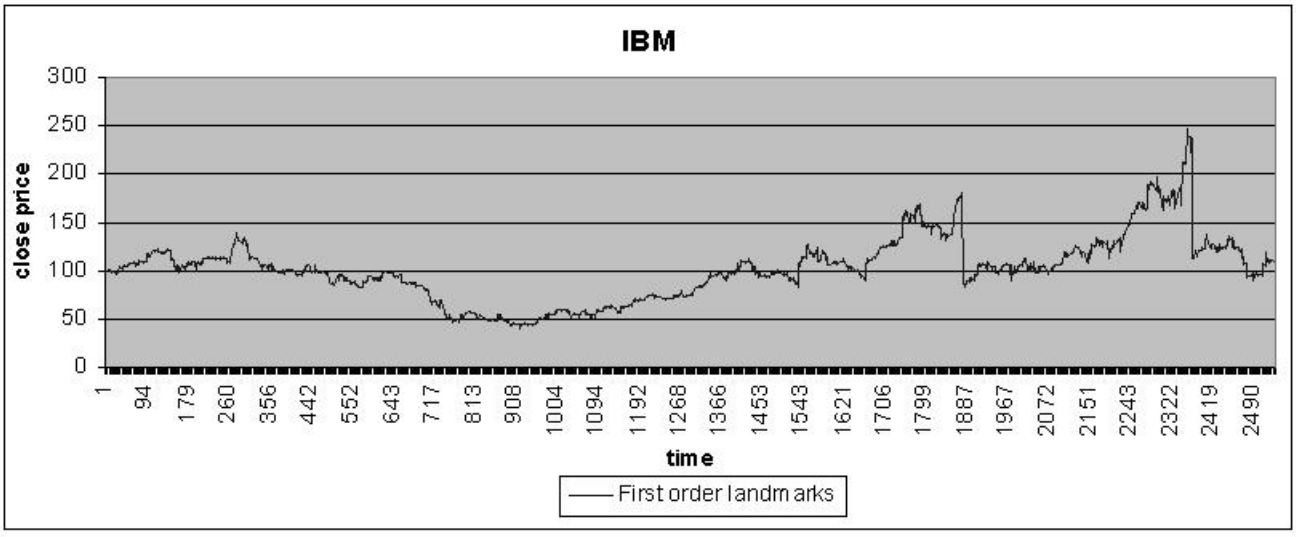

Figure 4.3c - IBM FO landmarks.

Let us define two important notions used extensively in the paper, namely uptrend and downtrend.

Let $\{(t_i, y_i)\}_{i=1}{}^{n}$ be a set of successive FO landmarks in the following definitions:

Definition [Uptrend] An **uptrend** is a succession of 3 first order landmarks (t_i, y_i), (t_{i+1}, y_{i+1}), (t_{i+2}, y_{i+2}), such that y_i and y_{i+2} are maximum points wrt the raw data and $y_i < y_{i+2}$.

Definition [Downtrend] A **downtrend** is a succession of 3 first order landmarks (t_i, y_i), (t_{i+1}, y_{i+1}), (t_{i+2}, y_{i+2}), such that y_i and y_{i+2} are minimum points and $y_i > y_{i+2}$.

In addition, either (t_i, y_i) and (t_{i+1}, y_{i+1}) satisfy MDPP condition and/or (t_{i+1}, y_{i+1}) and (t_{i+2}, y_{i+2}).

A succession of uptrends forms a bigger uptrend, as well as a succession of downtrends forms a bigger downtrend.

MDPP problem. The goal of the MDPP algorithm is to filter the FO time series using a conjunction of two conditions: $d < D$ and $p < P$ and to keep major maxima and minima points, also called turning points.

Our revised algorithm is inspired from the simplex method which was the first method developed to solve linear programming problems. Indeed, our MDPP algorithm, defined over a FO time series $\{(t_1, y_1),...,(t_n, y_n)\}$, can be seen as a union of some small linear programming subproblems. Every linear programming problem, LP_k is defined over a subseries $\{(t_i, y_i),...,(t_j, y_j)\}$ of the original time series such that all the vertices $[y_l, y_{l+1}]$ are convex, where l is from 1 to j-1.

A simplex method attempts to enclose the minimum/maximum inside an irregular volume defined by a simplex (= an n-dimensional convex volume bounded by $(n-1)$-dimensional hyperplanes and defined by $n+1$ linearly independent corners, e.g. a tetrahedron for $n = 3$). Thus, the simplex method assures that it is enough to go around the edges of the convex volume to find the optimal solution.

In the spirit of the simplex solution, our algorithm attempts to capture the trend of the time series by following only the maximum points during an uptrend, looking for the major maxima and, following only the minima points during a down trend, looking for the major minima. In this way, during the search for a maxima in an uptrend, all the minima points are ignored because the solution of the search cannot be a minimum point. Similarly, during the search for a minima in a downtrend, all maxima points are ignored.

We formalize the relation between two minima/maxima landmarks in an uptrend/downtrend by the following relation, called *Trend Relation.*

Definition [**Trend Relation** $\gg$] Given a time series FO $= [(t_i, y_i)]_{i=1..m}$ of first order landmarks in the result of findFo() algorithm over a time series TS, we define a *relation* $\gg$, called *Trend Relation* over FO such that:
$\forall$ i, j, with $i < j$, $(t_i, y_i) \gg (t_j, y_j)$ iff either

- (t_i, y_i) and (t_j, y_j) are both maxima and $y_i < y_j$ or
- (t_i, y_i) and (t_j, y_j) are both minima and $y_i > y_j$.

Figure 4.3(d) captures the Trend Relation $\gg$ among first order landmarks in a time series.

We are ready now to refine the definitions of major maxima and minima points.

Definition [**Major maxima**] A major maxima in a time series is a turning point from an uptrend to a downtrend. This point is also called a peak and has the property that it is the maximum point within the uptrend that preceedes this point.

Definition [**Major minima**] A major minima in a time series is a turning point from a downtrend to an uptrend. This point is also called a valley and has the property that it is the minimum point within the downtrend that succeeds this point.

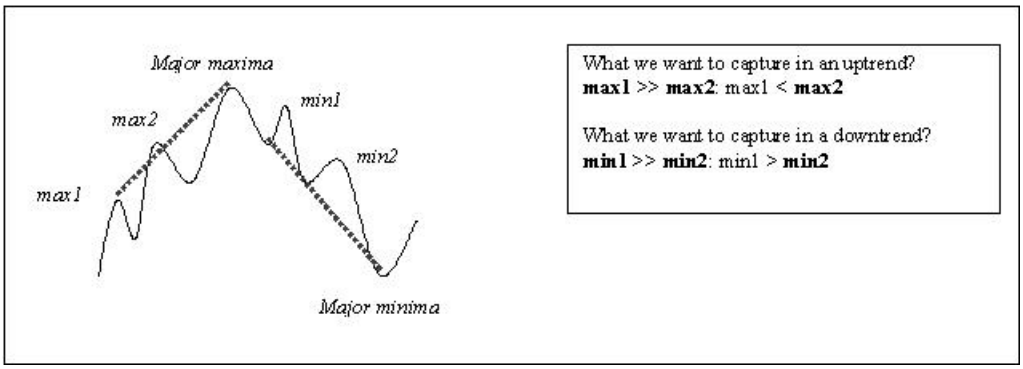

Figure 4.3d - The Trend relation $\gg$ defined over FO landmarks.

Our algorithm extends Perng's approach by using the same smoothing conditions ($d < D$ and $p < P$) in an attempt to filter out those insignificant landmarks. We have given a formalized definition of what important landmarks are and our goal is to find these landmarks during the smoothing process. The algorithm for computing the MDPP landmarks is given below and for simplicity and clarity, we do not insert the conditions for first and last points of the time series. Still we have to include them as borders of our final output.

Notation: we use DP(p_i, p_{i+1}) to represent the MDPP process of checking the conditions for distance and percentage smoothing parameters, between point i and point $i+1$.

$\downarrow DP(p_i, p_{i+1})$ and $\uparrow DP(p_i, p_{i+1})$ are two functions which have as arguments two consecutive first order points. $\downarrow DP(p_i, p_{i+1})$ takes as arguments only minimum first order points and $\uparrow DP(p_i, p_{i+1})$ takes as arguments only maximum first order points.

The Main function calls the $\downarrow DP(p_i, p_{i+1})$ function if the second point from the first order landmarks is a minimum point or it calles the $\uparrow DP(p_i, p_{i+1})$ function otherwise.

$\downarrow DP(p_i, p_{i+1})$ - *check the Trend relation between two local minima points*

if ($DP(p_i, p_{i+1})$ and $p_i \gg p_{i+2}$)		*//$p_i \gg p_{i+2}$ and MDPP conditions are true*
	$remove(p_i, p_{i+1})$;	*//remove small fluctuations*
	$\downarrow DP(p_{i+2}, p_{i+3})$;	*//follow the trend (downtrend)*
else		
	$keep(p_i)$	
	$\uparrow DP(p_{i+1}, p_{i+2})$;	*//change the trend (uptrend)*

$\uparrow DP(p_i, p_{i+1})$ - *check the Trend relation between two local maxima points*

if ($DP(p_i, p_{i+1})$ and ($p_i \gg p_{i+2}$))		*//$p_i \gg p_{i+2}$ and MDPP conditions are true*
	$remove(p_i, p_{i+1})$;	*//remove small fluctuations*
	$\uparrow DP(p_{i+2}, p_{i+3})$;	*//follow the trend (uptrend)*
else		
	$keep(p_i)$	
	$\downarrow DP(p_{i+1}, p_{i+2})$;	*//change the trend (to downtrend)*

MAIN () - *main function*

i = 2;		*//the second point from the time series*
if p_i is a local maxima point		
	$\uparrow DP(p_i, p_{i+1})$;	
else		
	$\downarrow DP(p_i, p_{i+1})$;	

The most important achievement of our algorithm is that it finds all global maxima and minima points.

Theorem 4.1 *No major maxima (minima) is lost in our MDPP algorithm.*

Proof During the smoothing process, MDPP algorithm keeps all the turning points from the time series, even if the mdpp condition is true, i.e. $d < D$ and $p < P$, and it smoothes the points from uptrends and the downtrend if they satisfy the mdpp condition. ∎

4.4 Assessments of MDPP algorithm

Before proceeding to check the effectiveness of the Mdpp smoothing algorithm in forecasting time series data, let us give some preliminary conclusions and results.

- MDPP algorithm finds all major maxima and minima landmarks. We consider that these landmarks have great importance in trading as every trader will want to buy/sell at every major valley/peak.
- Mdpp algorithm has an O(n) running time, where n is the length of the time series. Figure 4.4a shows the effect of the MDPP algorithm on one of the MSFT historical data, while using distance of 20 days and percentages of 0.7%.

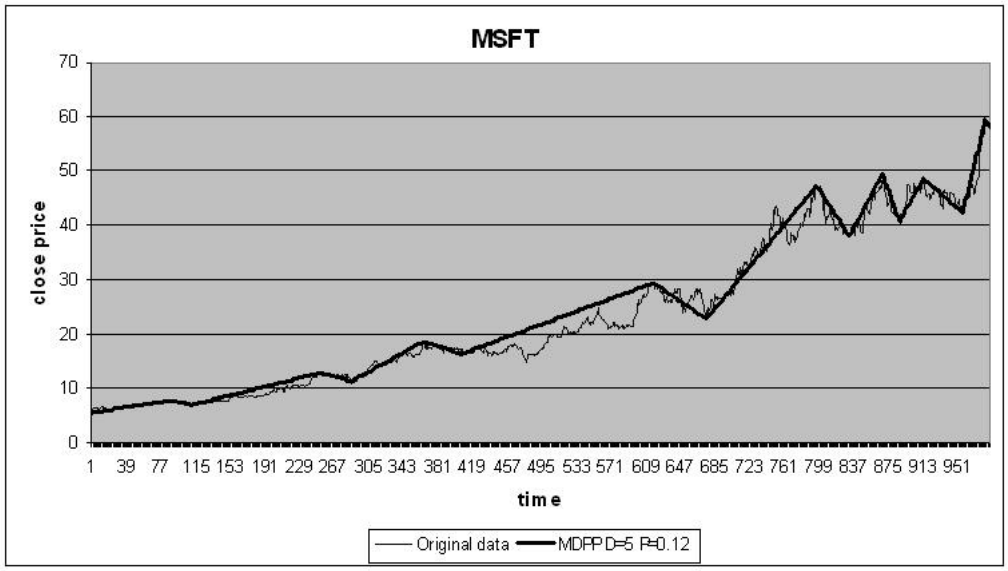

Figure 4.4a - MDPP iteration with distance 20 and percentage 0.7.

- We computed the normalized error between interpolated mdpp smoothing data and original data. The *normalized error* is measured by the *Normalized Function.*

  Definition [**Normalized distance function** δ] Given two sequences of length n, $X = \{x_i, ..., x_n\}$ and $Y = \{y_i, ..., y_n\}$, the **normalized distance function** δ is defined as:

$$\delta(X, Y) = \frac{1}{n} \sum_{i=1}^{n} \frac{|x_i - y_i|}{(x_i + y_i)/2}$$

It is found that applying the MDPP algorithm on time series data results in a great reduction in the number of points and still the normalized errors are small (see Table 4.4(t1)). In other words, we can use for example 2.3% of the original IBM data to represent the whole time series with only 3.98% normalized error.

So, when we vary the percentage and keep the distance constant, the number of the MDPP landmarks can be exponentially decreasing to some point where they are stabilizing (see Figure 4.4b).

D/P	2%	4%	6%	8%	10%	12%	14%
2	581/0.97%	515/1.1%	511/1.1%	509/1.15%	**509/1.15%**	**509/1.15%**	**509/1.15%**
4	437/1.18%	257/1.73%	223/1.83%	211/1.92%	**209/1.93%**	**209/1.93%**	**209/1.93%**
6	429/1.21%	213/2.03%	153/2.27%	129/2.48%	**121/2.57%**	**121/2.57%**	**121/2.57%**
8	421/1.22%	193/2.17%	137/2.38%	111/2.55%	101/2.71%	**93/2.97%**	**93/2.97%**
10	421/1.22%	191/2.19%	127/2.72%	99/2.99%	89/3.15%	**79/3.57%**	**79/3.57%**
12	421/1.22%	191/2.19%	125/2.76%	97/3.04%	85/3.21%	75/3.58%	73/3.68%
14	421/1.22%	189/2.2%	121/2.84%	93/3.12%	79/3.4%	69/3.76%	67/3.87%
16	421/1.22%	187/2.27%	117/2.97%	89/3.25%	75/3.52%	65/3.9%	61/4.12%
18	421/1.22%	183/2.3%	115/2.86%	87/3.14%	71/3.43%	61/3.76%	59/3.88%

D/P	16%	18%
2	**509/1.15%**	**509/1.15%**
4	**209/1.93%**	**209/1.93%**
6	**121/2.57%**	**121/2.57%**
8	**93/2.97%**	**93/2.97%**
10	**79/3.57%**	**79/3.57%**
12	**69/3.93%**	**69/3.93%**
14	**63/4.11%**	**63/4.11%**
16	57/4.37%	55/4.45%
18	**59/3.98%**	**59/3.98%**

Table 4.4(t1) The number of the remaining landmarks and the normalized error generated by the MDPP algorithm with different smoothing parameters D and P. The original data contains 2528 closing prices of IBM. The number of first ordered landmarks is 1227.

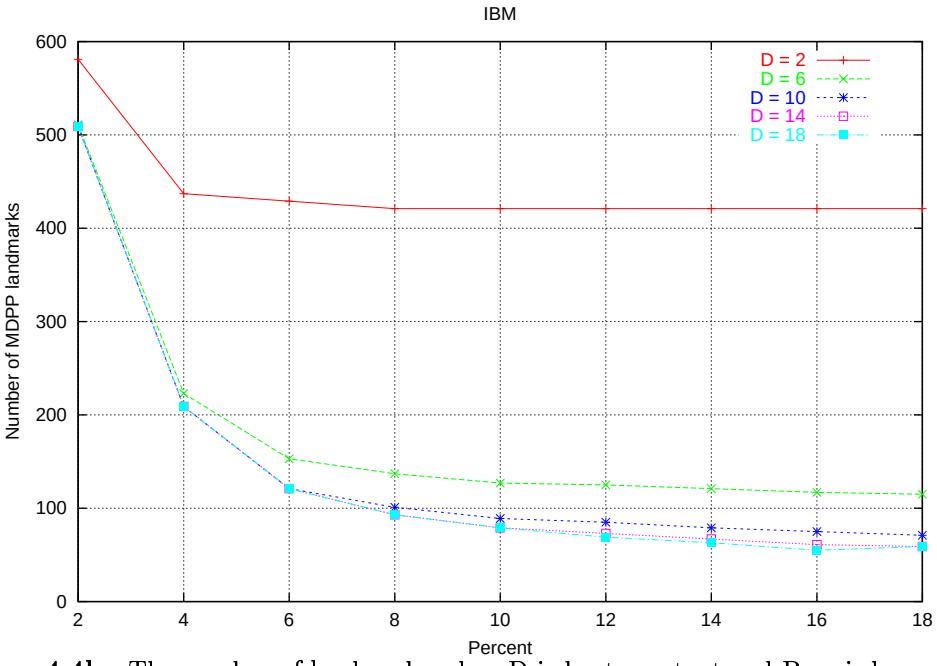

Figure 4.4b - The number of landmarks when D is kept constant and P varied.

5 Non-parametric Kernel Regression

In some financial applications, we may not have sufficient knowledge to pre-specify the nonlinear structure between two variables Y and X. In other applications we may wish to take advantage of the computing facilities and computational methods to explore the functional relationship between Y and X. These considerations lead to the use of nonparametric methods. Use of nonparametric methods, however, are not without any cost. They are highly data-dependent and can easily result in over fitting. Let us discuss first about smoothing estimators in general and then about the most well known nonparametric method, *Kernel regression*. Other nonparametric methods that are not discussed here include local least squares estimation and neural network.

5.1 Smoothing Estimators

The essence of nonparametric methods is *smoothing*. In addition to possessing certain statistical optimality properties, smoothing estimators are motivated by their close correspondence to the way human cognition extracts regularities from noisy data. Therefore, they are ideal for our purpose [6].

We first begin in a quantitative manner by stating that prices $\{P_t\}$ in a time series at time t can be expressed as follows:

$$P_t = m(X_t) + \epsilon_t, \qquad t = 1, 2, ..., T$$

where $m(X_t)$ is an arbitrary fixed non-linear function of a variable X_t and $\{\epsilon_t\}$ is white noise [4].

The $m(X_t)$ series can be the smoothed price series [5].

Next, we further explain the concept of weighted average in detail. Given a time series, suppose we want to estimate $m(.)$ at a particular time t_0 where $X_{t_0} = x_0$ by notation. If we assume that the function $m(.)$ is reasonably smooth, then the P_t values at the observations near x_0 should be close to $m(x_0)$. This means that if $m(.)$ is rather smooth, then $m(x_0)$ will be almost consistent in a small neighborhood around x_0. As a result, $m(.)$ can be estimated by averaging the P_t values of those X_t near x_0. As X_t approaches to x_0, the average of their P_t values tends closer towards $m(x_0)$. Hence, a weighted average of the P_t values tends closer towards $m(x_0)$.

Thus, a weighted average of the P_t values is used, where lesser wights are applied as X_t shifts away from x_0. This weighted average of estimating $m(x)$ can be expressed as:

$$\hat{m}(x) = \frac{1}{T}\sum_{t=1}^{T} \omega_t(x) P_t \qquad \textbf{(1)},$$

where the weights $\{\omega_t(x)\}$ are bigger for those prices P_t with X_t nearer to x, and smaller for those further away. However, there is a tradeoff between variance and bias. If we select too large a region around x to compute the average, the weighted average will be too smooth and hence becomes overly biased. In conclusion, it is very important to select the weights $\{\omega_t(x)\}$ carefully to balance these two issues.

5.2 Kernel regression

Kernel regression is perhaps the most commonly used nonparametric method in smoothing. The weights here are determined by a *kernel*, which is typically a probability density function (pdf), $K(t)$, that satisfies:

$$K(x) \geq 0, \int K(u)\, du = 1$$

This function $K(x)$ is also known as *kernel*. We can change the range of the kernel by varying it with respect to a parameter h (also known as *bandwidth*), where $h > 0$.

The rescaled kernel becomes $K_h(u) = \frac{1}{h}K(\frac{u}{h})$, $\int K_h(u)\, du = 1$. The weight function, $\omega_t(x)$, can now be defined as:

$$\omega_{t,h}(x) = \frac{K_h(x - X_t)}{g_h(x)}, \text{ where } g_h(x) = \frac{1}{T}\sum_{t=1}^{T} K_h(x - X_t). \qquad \textbf{(2)}$$

The bandwidth h controls the scale of the region around each of the X_t values for averaging. In other words, if the bandwidth is small, then the average will be calculated with respect to a relatively small region around the X_t values. Conversely, if h is large, the average will be computed over a larger region around the X_t values. So, the degree of averaging actually depends on the bandwidth h, and thus selecting the suitable bandwidth is an important issue. By substituting equation (2) into equation of the weighted average (1), we have the kernel non-parametric estimator $\hat{m}_h(x)$ of $m(x)$:

$$\hat{m_h}(x) = \frac{1}{T}\sum_{t=1}^{T}\omega_{t,h}(x)P_t = \frac{\sum_{t=1}^{T}K_h(x-X_t)P_t}{\sum_{t=1}^{T}K_h(x-X_t)}.$$

This $\hat{m_h}$(x) is also often refereed as the *Nadaraya-Watson* kernel estimator [7, 10]. In practice, many choices are available for defining Kernel $K(x)$. However, for theoretical and practical considerations, we choose a Gaussian kernel:

$$K_h(x) = \frac{1}{h\sqrt{2\pi}}\exp\left(\frac{-x^2}{2h^2}\right) \qquad \textbf{(3)}$$

5.2.1 Bandwidth Selection

There are several approaches for automating the choice of bandwidth selection [19]. The method that we will use for bandwidth selection is the *leave-one-out cross-validation*. In this method, first one observation (x_j, y_j) is left out and the remaining T-1 data points are used to obtain the following smoother at x_j:

$$\hat{m_{h,j}}(x_j) = \frac{1}{T-1}\sum_{t=j}\omega_t(x_j)y_t,$$

which is an estimate of y_j. Second, perform Step-1 for $j = 1, ..., T$ and define the function:

$$CV(h) = \frac{1}{T}\sum_{j=1}^{T}\left[y_j - \hat{m_{h,j}}(x_j)\right]^2 W(x_j),$$

where $W(.)$ is a non-negative weight function that can be used to down-weigh the boundary points if necessary. Decreasing the weights assigned to data points close to the boundary is needed because those points often have fewer neighboring observations. The function *CV(h)* is called the *cross-validation function* because it validates the ability of the smoother to predict $\{y_t\}$ for $t = 1, ..., T$. One then chooses the bandwidth h that minimizes the *CV(.)* function.

One interesting finding in Andrew Lo's paper, [15], is that the bandwidths obtained from minimizing the cross validation function are generally too large for their application to technical analysis and the fitted functions were too smooth. In other words, the cross-validation determined bandwidth places too much weight on prices far away from any given time t, inducing too much averaging and discarding valuable information in local price movements. In [15],they propose an acceptable solution of using a bandwidth equal with $0.3*h$, where h minimizes *CV(h)*.

6 Forecasting by Smoothing

According to one of the classic references in technical analysis, "the whole purpose of charting... is to identify trends in the early stages of their development for the purpose of trading in the direction of those trends" [11]. This activity become popular after John Magee and R.D Edwards published the first of many editions of *Technical Analysis of Stock Trends* in 1948.

6.1 About Chart Pattern Language (CPL)

Chart Pattern Language (CPL), proposed by Saswat Anand in his MSc thesis in 2001 [16], is a high-level pattern definition language to facilitate pattern discovery process. The language enables financial analysts to do the following tasks:

1. Define patterns with fuzzy constraints. Through incremental addition of fuzzy constraints to a pattern, the user is able to refine patterns iteratively.

2. Reuse patterns. Compose complex patterns are built by composing simpler patterns and refining them by incrementally adding further constraints.

3. Search for patterns in price history. The discovery process of such patterns can be extremely laborious and technically challenging in the absence of a high level pattern definition language.

By embedding Chart Pattern Language within Haskell, the user can reap benefits from various nice features of Haskell. Specifically,

- Haskell's strong type system infers the type of pattern definition automatically. This frees programmers, who are financial analysts by profession, from the mundane task of declaring variables and specifying types - a task which they are not comfortable with nor enjoy doing.
- High-order functions are used extensively throughout the system to create a natural and concise syntax for the language. CPL has been designed with the aim of making it similar to spread-sheet formula language, which financial analysts are familiar with.
- Searching for patterns in price history involves multiple constraints satisfaction, which has a worst case exponential running time. Lazy Evaluation plays a crucial role by automatically avoiding unnecessary computation during searching for patterns.

In CPL, a pattern is featured by a sequence of landmarks. There are two important basic patterns, *up* and *down* patterns.

The price movement from a time a to a time b constitutes an *up* pattern if $high_t < high_b$, for all t $\in [a, b)$ and $low_t < low_a$ for all t $\in [a, b)$.

The price movement from a time a to a time b constitutes a *down* pattern if $low_t > low_b$, for all t $\in [a, b)$ and $high_t < high_a$ for all t $\in [a, b)$.

An up pattern describes the price movement between two landmarks points which indicate an upward trend in the stock prices. A down pattern reflects a downward trend in the stock prices between two landmarks.

Complex patterns are a composition of a series of primitive patterns (sub-components) which are subject to some constraints. A constraint function specifies a list of global constraints about the pattern. These constraints are usually expressed in terms of the pattern's landmarks and sub-components. The constraint function is evaluated with respect to a pattern instance and it is not wrong to view the constraint function as a part of a pattern, since the it applies consistently to *all* instances of the pattern.

In short, CPL provides a high-level platform upon which analysts can define and search pattern instances easily and without any programming expertise. A *pattern instance* is differentiated from a pattern in that it is an occurrence of the pattern in the price history. In order to find all the patterns instances of a particular pattern, CPL searches for all possible cases using the landmark detail of the pattern, such that these cases satisfy the pattern constraints.

6.2 Head and Shoulder Pattern

Much less academic attention has been paid to the use of *technical signals* based on *price patterns*, despite the fact that these are widely used by practitioners. Osler and Chang [14] examine the profitability of using the *"head-and-shoulder"* pattern in the foreign exchange market to predict changes of trend, and find evidence of excess returns for some currencies. Lo et al. [15] develop a pattern detection algorithm based on kernel regression. They apply this algorithm to identify a variety of technical price patterns including "head-and-shoulder" in the U.S stock market over the period 1962-1996. They find statistical evidence that there is potentially useful information contained in most of the patterns they consider.

One of the difficulties that an academic investigator face in assessing the predictive power of price patterns is that the characterization of patterns is sometimes ambiguous and there may be disagreement among technical analysts themselves. The individual realization of patterns can indeed vary: "Charts patterns are like music. There are scores of variations on a theme" [4]. We have chosen for our investigation the *head-and-shoulder* pattern because there is a very general concesus on the importance features of this pattern, and it is also agreed that it is one of the most reliable technical indicators [8, 2].

Modifications

Head-and-Shoulder (HS) pattern is a reversal pattern and it is most often seen in upward trends. This pattern usually occurs when the market begins to slow down and the forces of supply and demand are generally considered balanced. With reference to Figure 6.2a, sellers enter the market when the price reaches the first peak at the *left shoulder* and then the downside is checked (beginning of the neckline). When this happens, buyers return to the market and ultimately push through to another new and higher peak, the *head*. However, this quickly reversed and the downside is tested once more, so we have the continuation of the neckline. Tentative buying emerges again and the market peaks up again, but fails to replace the previous peak. This peak is the *right shoulder*. Eventually, buying dries up and the market tests the downside yet again. New sellers come in and previous buyers exit the market. The head-and-shoulder pattern is complete when the market breaks the neckline.

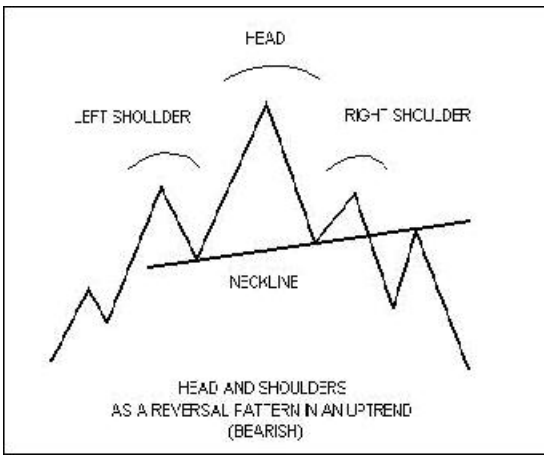

Figure 6.2a. Head-and-Shoulder Pattern [9]

Our methodology for defining HS pattern differs from Lo et al. [15] and Savin [9] in a couple of aspects. Let us describe further the HS pattern as it is in our CPL language.

In CPL, the head-and-shoulder pattern consists of six primitive patterns [16]. Its corresponding seven landmarks are labeled from e_0 to e_6 in Figure 6.2b, such that e_1 is the left shoulder, e_3 is the head and e_5 is the right shoulder.

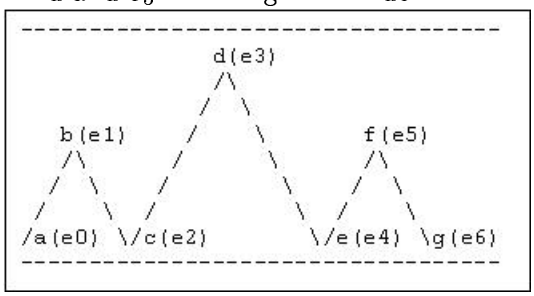

Figure 6.2b. Head-and-Shoulder Landmarks

In Lo et al. [15], the head-and-shoulder pattern is subjected to the following restrictions.

(R_1) e_1 is a maximum.

(R_2) $e_3 > e_1$.

(R_3) $e_3 > e_5$.

(R_4) e_1 and e_5 are within 1.5% of their average, i.e.,

$$max_i|e_i - avg| \leq 0.015 * avg, \text{ i = 1,5, where } avg = \frac{e_1 + e_5}{2}$$

(R_5) e_2 and e_4 are within 1.5% of their average, i.e.,

$$max_i|e_i - avg| \leq 0.015 * avg, \text{ i = 2,4, where } avg = \frac{e_2 + e_4}{2}$$

The restrictions (R_1), (R_2) and (R_3) are straightforward by seeking to maintain the geometrical shape of the head-and-shoulder pattern. The effect of the restrictions (R_4) and (R_5) is to limit the height difference between the left and right shoulder (i.e., e_1 and e_5) and the left and right trough (i.e., e_2 and e_4) to within 1.5%.

On the other hand, Savin et al. [9] suggests modifying the restrictions (R_4) and (R_5) as follows.

(R_{4a}) e_1 and e_5 are within 4% of their average, i.e.,

$$max_i|e_i - avg| \leq 0.04 * avg, \text{ i = 1,5, where } avg = \frac{e_1+e_5}{2}$$

(R_{5a}) e_2 and e_4 are within 4% of their average, i.e.,

$$max_i|e_i - avg| \leq 0.04 * avg, \text{ i = 2,4, where } avg = \frac{e_2+e_4}{2}$$

(R_{4a}) and (R_{5a}) actually increase the height difference between the left and right shoulders and between the left and right troughs to within 4% of their average.

The neckline (the line joining e_2 and e_4) is allowed to be steeper. In addition, Savin [9] impose four additional restrictions because they argue that restrictions (R_1) to (R_5) do not encompass all the features of a head-and-shoulder pattern, relevant to technical analysts. They are as follows:

(R_6) $\frac{\frac{[(e_1-e_2)+(e_5-e_4)]}{2}}{e_3-\frac{(e_2+e_4)}{2}} \leq 0.7$

(R_7) $\frac{\frac{[(e_1-e_2)+(e_5-e_4)]}{2}}{e_3-\frac{(e_2+e_4)}{2}} \geq 0.25$

(R_8) $\frac{[(e_3-\frac{(e_2+e_4)}{2}]}{e_3} \geq 0.03$

(R_9) $max_{i=1}|(t_{i+1} - t_i) - \bar{t}| \leq 1.2 * \bar{t}$, $i = 1, ..., 4$ where t_i is the date on which e_i occurs and $\bar{t} = \sum_{i=1}^{4} \frac{t_{i+1}-t_i}{4}$

(R_6) sets the upper bound on the proportion of the average height of the shoulders to the height of the head from the neckline to be 70 percent. (R_7) is the corresponding lower bound which is 25 percent. Specifically, (R_6) and (R_7) combined with (R_{4a}) and (R_{5a}) mainly rules out cases where the height of the shoulders is very large or very small compared to the height of the head. (R_8) rules out cases where the height of the head from the neckline is less than 3 percent of the stock price. Finally, (R_9) rules out extreme horizontal asymmetries in the head-and-shoulder patterns [9].

These restrictions (R_6) to (R_9) are calibrated using eleven examples of head-and-shoulder patterns reported in Bulkowski, T.N. [2, 1], that are completed within 63 trading days. In conclusion, (R_{4a}), (R_{5a}), (R_6), (R_7), (R_8) and (R_9) are referred to as the Bulkowski restrictions.

In addition to Lo's restrictions (R_1 to R_3) and Bulkowski restrictions, we further impose another restriction for testing purposes:

(R_{10}) t_1, t_2,..., $t_7 \in [1, T - 60]$, where t_i is the date on which e_i occurs and T is the length of the price series.

This is to ensure that the last landmark of any pattern instance cannot fall within the last 60 days of the price series. We do this because we need to use the next 60 days of a pattern instance to test for its returns.

Figure 6.2c provides the definition of the Head-and-Shoulder using our CPL.

Another modification is related with the spanning windows, from 63 proposed by Bulkowski to 400. That is, the price is divided into successive, overlapping windows of span = 400 trading days, where the difference between the limits of two adjacent windows is one trading day. The motivation for using windows is twofold. One is that it approximatevely mimics the way in which traders anayze the data. If windows did not overlap then the pattern algorithm would not detect any pattern initiated in one window and completed in the next. In contrast, traders use all historical price data as time unfolds. The other is that it automatically constrains the maximum length of the HS pattern. The purpose of using 400 instead of 63 is to verify the result obtained by Bullkowski accommodating the average completion time of an HS pattern.

```
pattern_hns = up >.> ( down >.> ( up >.> ( down >.> ( up >.> down ))))

hns = pattern_hns ? cons_hns
cons_hns :: forall a b c d. (LMS a b, Ind b c, Compare c d) => a -> [d]
cons_hns = \p->
      let  [a,b,c,d,e,f,g] = lms p

      e0 = close a
      e1 = close b
      e2 = close c
      e3 = close d
      e4 = close e
      e5 = close f
      e6 = close g
      x = (e1 + e5) / 2
      x1 = abs (e1-x)
      x2 = abs (e5-x)
      y = (e2 + e4) / 2
      y1 = abs (e2 - y)
      y2 = abs (e4 - y)
      z1 = (((e1-e2)+(e5-e4))/2) / (e3-(e2+e4)/2)
      z2 = (e3-(e2+e4)/2)/e3
      z3 = cast (f - b) / 4
      m = e4 - e2
      g0 = m*(cast(g-c)/cast(e-c)) + e2

      in [
            -- (R2) --
            e3 > e1,
            -- (R3) --
            e3 > e5,
            -- (R4a) --Allowable difference btw the PRICES of e1 and e3
            x1 <= 0.04 * x,
            x2 <= 0.04 * x,
            -- (R5a) -- Allowable difference btw the PRICES of e2 and e4
            y1 <= 0.04 * y,
            y2 <= 0.04 * y,
            -- (R6) -- Upper bound
            z1 <= 0.7,
            -- (R7) -- Lower bound
            z1 >= 0.25,
            -- (R8) --
            z2 >= 0.03,
            -- (R9) -- Rules out extreme horizontal asymmetries
            abs (cast(c-b)-z3) <= 1.2 * z3,
            abs (cast(d-c)-z3) <= 1.2 * z3,
            abs (cast(e-d)-z3) <= 1.2 * z3,
            abs (cast(f-e)-z3) <= 1.2 * z3,
            -- (R10) -- -- Ensure 60 days of slack for testing purposes
            close (g+60+3) > 0
      ]
```

Figure 6.2c. CPL code: Head-and-Shoulder definition in Haskell

6.3 Testing Methodology

Our main aim is to assess the effectiveness of our MDPP smoothing algorithm in finding head-and-shoulder pattern instances in stock prices. In this chapter, we first describe the data sets used for

testing and the procedures for calculating the return conditional on detecting a head-and-shoulder pattern instance. In the second part, we present our tests with the MDPP and the Kernel regression smoothing techniques and the results of the search for head-and-shoulder pattern instances.

6.3.1 Stock Market Data

We conducted experiments for 362 different workloads which are company's data sets selected from S&P 500 index, as they are traded more frequently and enjoy ample liquidity. This is a necessary condition for our tests as prices are indicated more accurately.

There are a total of 2528 trading days from the period 1st January 1990 to 31st December 1999 for each of the 362 companies. Therefore, the length of each price series is 2528. For each individual company, we have their open, high, low, close prices and the volume of each trading day. So, we have actually 2528 sets of open, high, low, close prices and volume. However for simplicity, we only use close prices in the testing.

6.3.2 Procedure for Calculating Excess Returns

A good indicator of the effectiveness of head-and-shoulder pattern is whether there is any excess return to be earned, consistently. Instead of taking just the stock return, we need to take into account the opportunity cost of the stock investment which is putting the money in a bank, for example. If an investor earns 1% stock returns over a year but the bank is paying out interest of 2% for that year, the investor would have been better off by putting his money in the bank instead. Therefore, we consider a stock's excess return as the difference between the stock's continuously compounded return and the continuously compounded risk-free return.

For the risk-free rate, we use the daily U.S.-3 month Treasury Bill yielded for the corresponding period from 1st January 1990 to 31st December 1999.

For each head-and-shoulder pattern instance detected, we calculate the continuously compounded return for that stock over the subsequent 20, 40 and 60 trading days. We choose those specific intervals because in technical trading practices these are considered substantially longer horizons [2]. However, there is no clear consensus on one appropriate horizon.

The *Continuously Compounded Return* ($r_{i,c}$) for a stock is defined as:

$$r_{i,c} = \ln \frac{P_{i+c}}{P_i}$$

where
i = last day of the pattern instance, $i \in [1, 2528]$
$c = 20, 40, 60$
P_j = close price for day j.

We then calculate the continuously compounded risk-free return over the same holding period of 20, 40 and 60 subsequent trading days for that stock. The *Continuously Compounded Risk-Free Return* ($r_{rf,i,c}$) is calculated as

$$r_{rf,i,c} = \exp(\frac{1.4*c}{365} * \ln(1 + r_{rf,i})) - 1$$

where
i = last day of the pattern instance, $i \in [1, 2528]$
c = 20, 40, 60
$r_{rf,i}$ = U.S. 3-month Treasury Bill yield for day i.

We then calculate the stock's *Excess Return ($r_{e,i,c}$)* by subtracting the continuously compounded risk-free return from the stock's continuously compounded return, i.e.,

$$r_{e,i,c} = r_{i,c} - r_{rf,i,c}$$

where i = last day of pattern, $i \in [1, 2528]$
c = 20, 40, 60.

For each pattern instance found in a particular stock, all these three returns are computed over the same period of 20, 40 and 60 subsequent trading days after the last day of the pattern instance formation. Hence if n pattern instances are found from the data set, we will obtain n sets of excess returns for the three holding periods.

Next we compute the mean excess returns for the three holding periods from the n pattern instances found in the data set. We also compute their standard deviations and the t-values. From their t-values, we put the results to a t-test for 20, 40 and 60 trading days. The null hypothesis (H0) to be used for the t-test is:

H0 = the true mean for the excess return of the pattern instance is zero.

Depending on the predicted trend of the pattern (upward trend or downward trend), the alternative hypothesis will be that the true mean is not equal to zero. In our case, since we are testing for head-and-shoulder pattern which is a reversal of an upward trend, therefore the alternative hypothesis (H1) would be:

H1 = the true mean for the excess return of the pattern instance is less than zero.

If the null hypothesis is rejected, it means that the actual sign of the pattern instance corresponds to the predicted one (i.e., negative) in the alternative hypothesis. *A negative excess return implies that a short sale is on average a profitable trading strategy.* We can then compare the magnitude of the returns to those of other successful pattern instances.

6.4 Descriptive Findings

6.4.1 MDPP Results

We conduct out tests in a few steps. First, we smooth the datasets for all of the 362 companies with the MDPP algorithm. The distance and percentage smoothing parameters range from [2, 48] with

an increment of 1 and [0.02, 1] with an increment of 0.01, respectively. However, we observed that the results for a percentage ≥ 0.20 are relatively constant. Therefore, we concentrate our tests for a percentage smaller than 0.20. In the second step, we search for head-and-shoulder instances using CPL. In the last step, we subject these head-and-shoulder instances to the t-test. The formula that we use for the t-test is:

$$\text{t-test} = \frac{x}{\frac{\sigma}{\sqrt[2]{n}}}$$

where x = mean return;
σ = standard deviation;
n = sample size.

Let us describe further our results:

- As the distance and percentage get larger, highly significant negative excess returns are yielded for 20, 40, 60 trading days, although the excess returns are dramatically decreased around distance of 32 days and large P values, i.e 0.5%, 0.7%. We assess the significance of the negative excess returns by their coresponding t-values. As their t-values are considerably big in absolute value, the null hypothesis H_0 that the true means are zero is rejected under 95% confidence interval. This supports the conjecture that the head-and-shoulder patterns have relatively strong predictive powers.

  An interesting observation is that the mean excess return and the mean length of the pattern instances are almost constant for large P values (greater than 0.1) and distance less than 12 days.

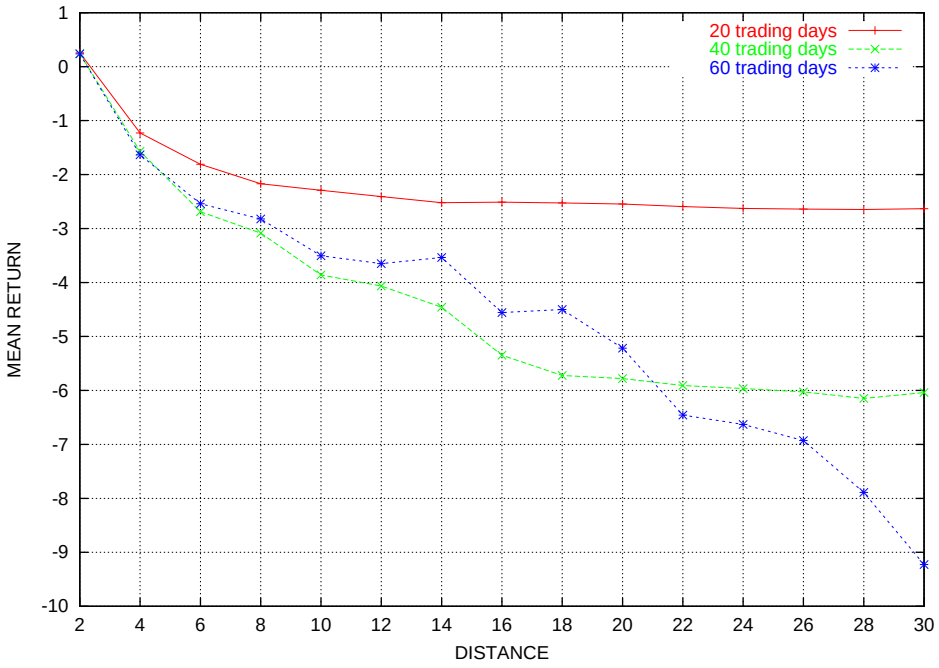

Figure 6.4.1a. Mean Returns with increasing D for 20,40 and 60 trading days

- The values of the mean excess returns and the t-values tend to be more and more negative, exponentially decreasing as the parameters D and P increase. For small percentage values (P = 0.05), the curves of the mean return and t-value are smooth but as the percentage is increasing, the curves become more ragged and we can observe large fluctuations in the mean return and t-values.

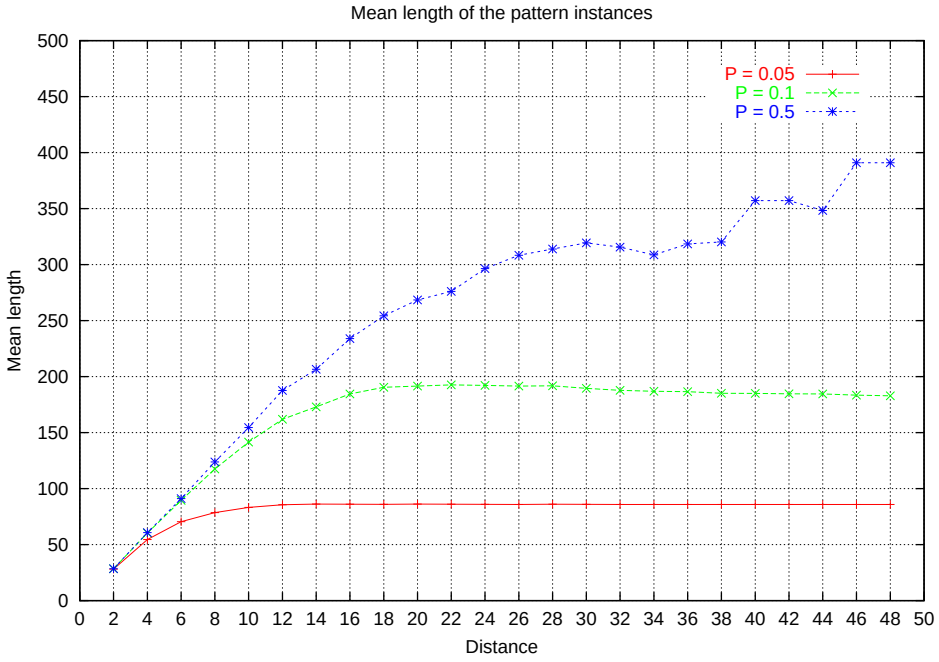

Figure 6.4.1b. Mean length of the pattern instances for 40 trading days.

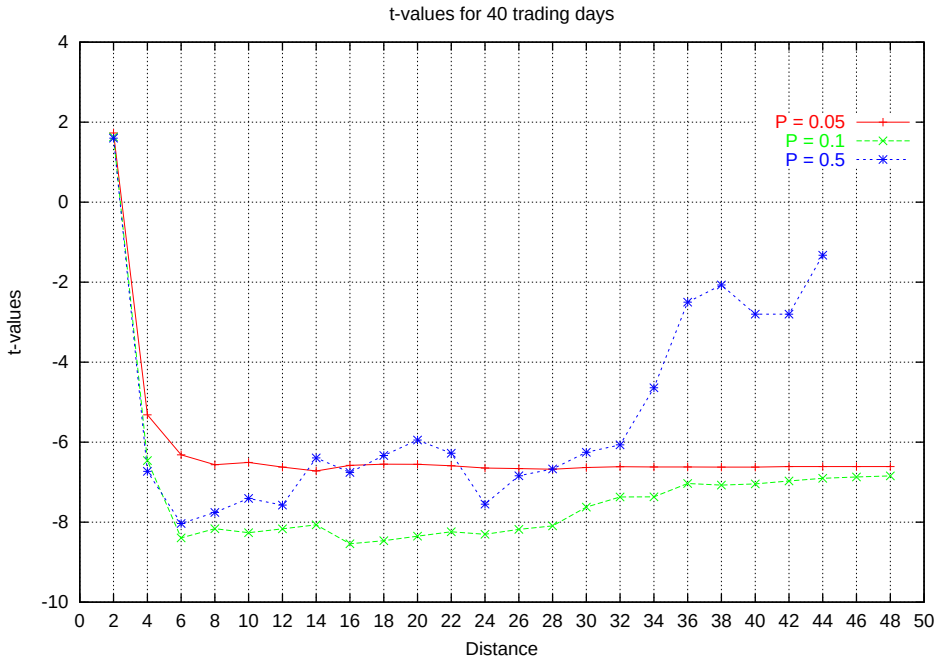

Figure 6.4.1c. t-values of the pattern instances for 40 trading days.

- We want to find further what is the success rate of the head-and-shoulder pattern instances found in all 362 companies. The success rate is the measure of the pattern instances with true means less than zero to the total number of pattern instances found. From our experiments, we conclude that for a small percentage, the mean number of instances is decreasing exponentially with distance from 1 to around 10 and then it's stabilizing.

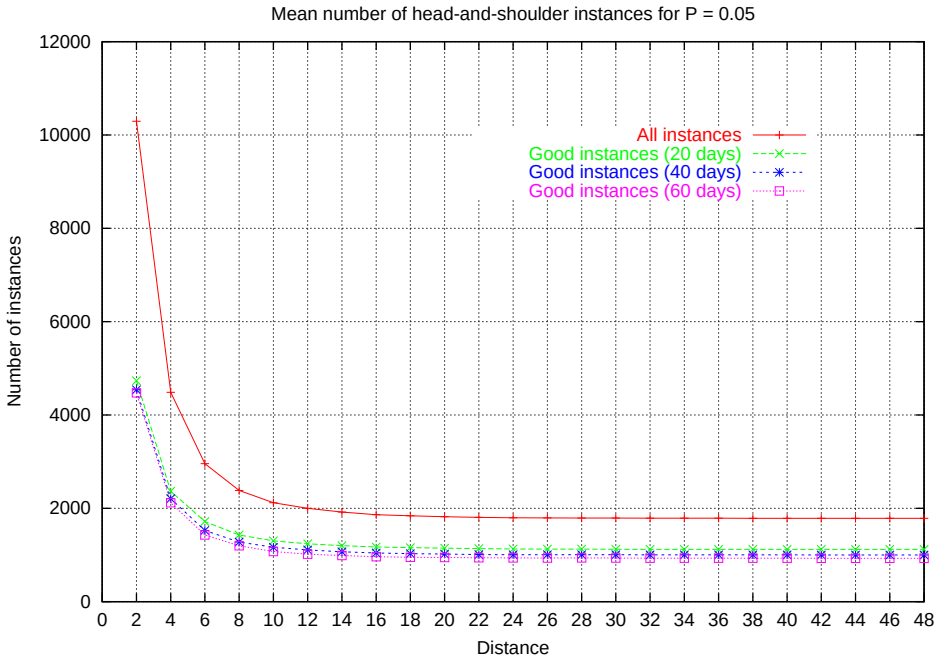

Figure 6.4.1d. The number of good instances for P = 0.05.

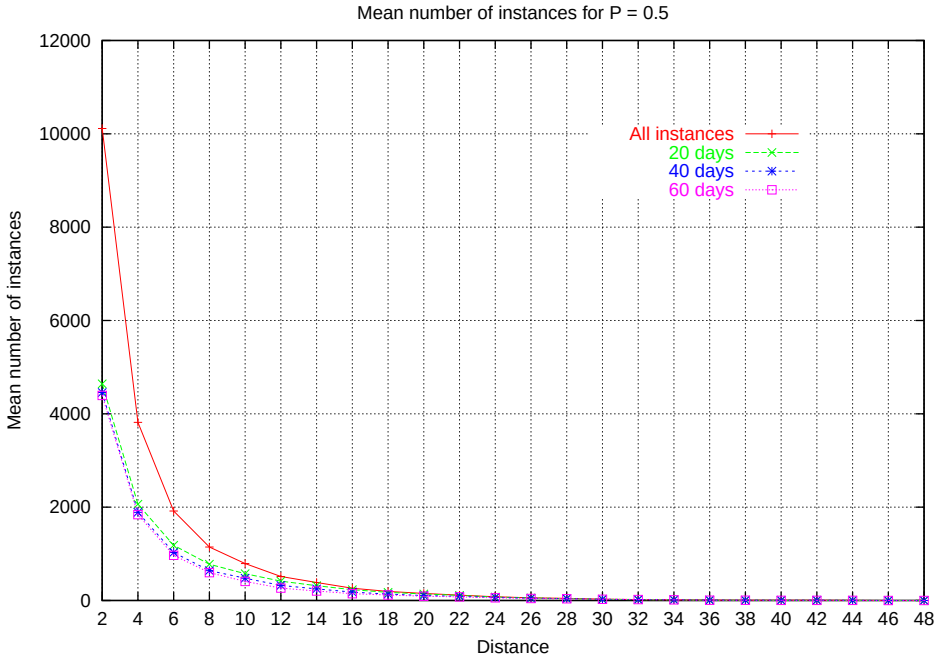

Figure 6.4.1e. The number of good instances for P = 0.5.

If we compare the success rates for a small percentage and a big percentage (i.e P = 0.05 and P = 0.5), we can conclude that for the smallest percentage, the success rate is low compared

with the success rate obtained with a bigger percentage and rarely exceeds 70%. Moreover, a big percentage goes to 100%. However, this does not imply that we have a better result since the number if instances is decreasing exponentially with the decrease of P. Still, it remains an open issue that the pattern instances found for a larger value of the P, often tend to be predictive with true means less than zero, hence the success rate will be high.

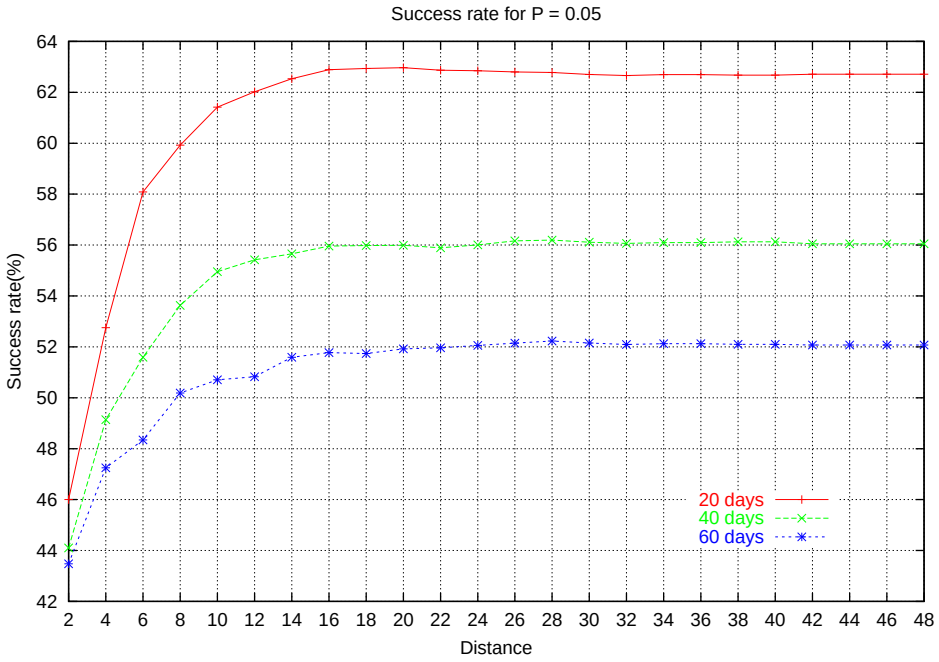

Figure 6.4.1f. The success rate of the head-and-shoulder instances for P = 0.05.

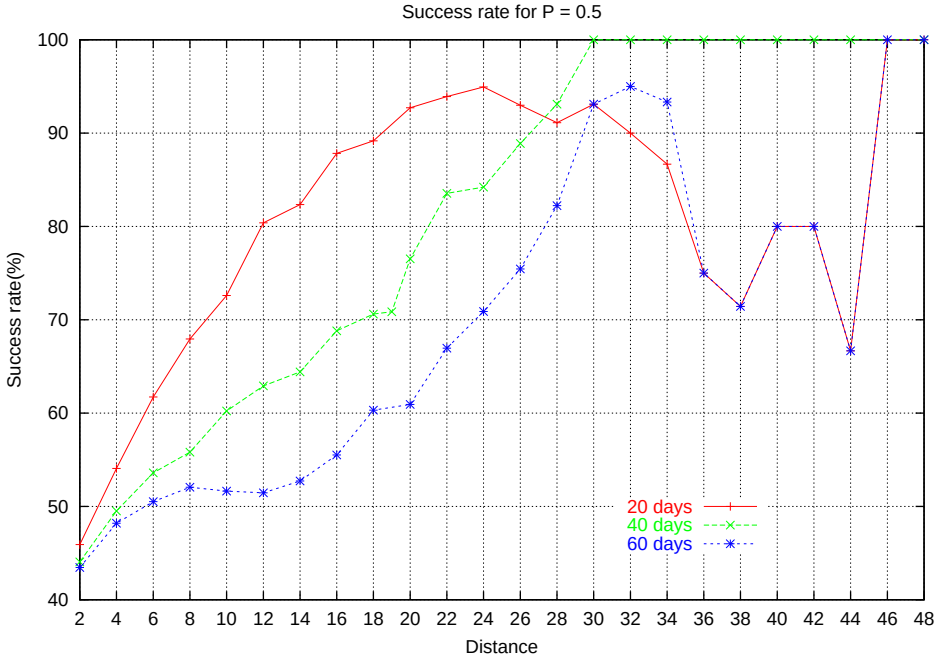

Figure 6.4.1g. The success rate of the head-and-shoulder instances for P = 0.5.

For bigger percentage values, and with distance less than 30 days, an interesting fact is that

a good success rate can be obtained for 20 trading and a low success rate can be obtained for 60 trading days. The success rate for 40 trading days falls between 20 and 60 trading days. When distance is around 30 days, the success rates for 20, 40 and 60 are almost similar. We underline the result obtained for a distance greater than 34 days, when we obtain 100% sucess rate for 40 trading days and equal results, still good ones, for 20 and 60 trading days.

6.4.2 Kernel Results

Four our Kernel regression tests, we use values of the bandwidth h from 0 to 10 with an increment of 0.5. The true means for the excess returns and their corresponding t-values, for all values of h, calculated at 20, 40 and respectively 60 days, are shown in Figure 1 and 2 below.

- The results show that there are significant negative excess returns for all the bandwidths except for h=0.5. More specifically, the excess returns are are almost equal for values of the bandwidths smaller than 4.5. For bandwidth grater than 4.5, we observe a distribution of the mean returns as follows: for 20 trading days, the mean return oscilates around -6 value, for 40 trading days, around -5 value and for 60 trading days, around -4 value. So, the null hypothesis H_0 is rejected under 95% confidence interval.

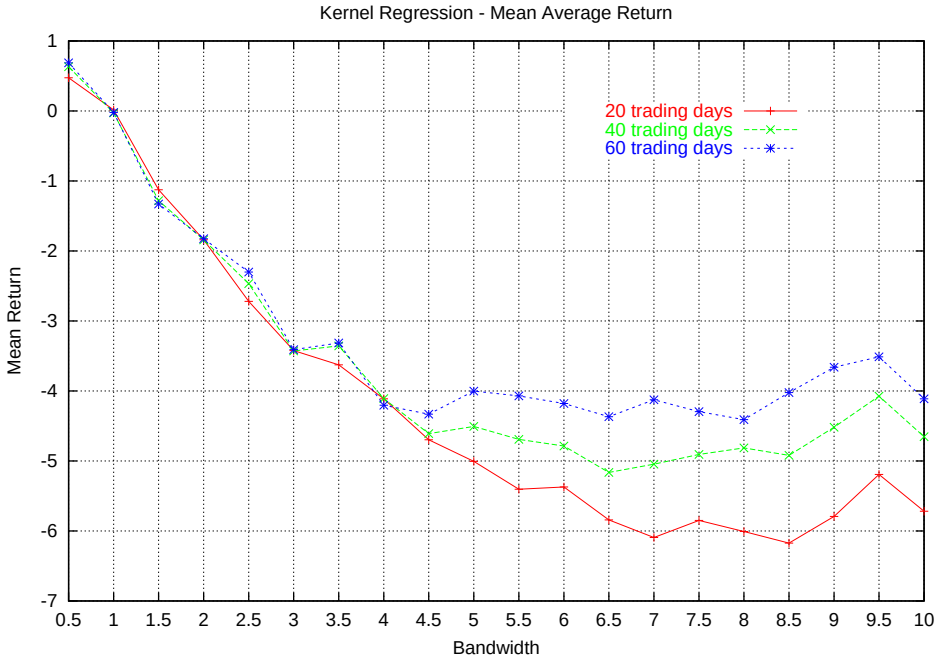

Figure 6.4.2a. The average mean returns using Kernel Regression, for 20, 40 and 60 days.

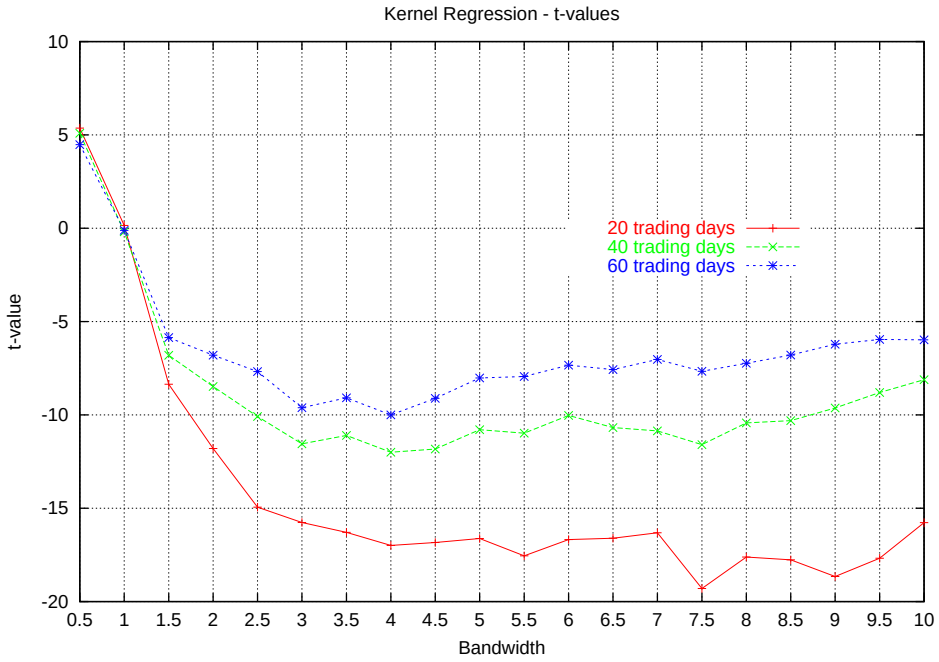

Figure 6.4.2b. The t-values using Kernel Regression, for 20, 40 and 60 days.

- The number of the "head-and-shoulder" instances found when smoothing with Kernel Regression is decreasing exponentially until it is stabilizing around 250 number of the instances.

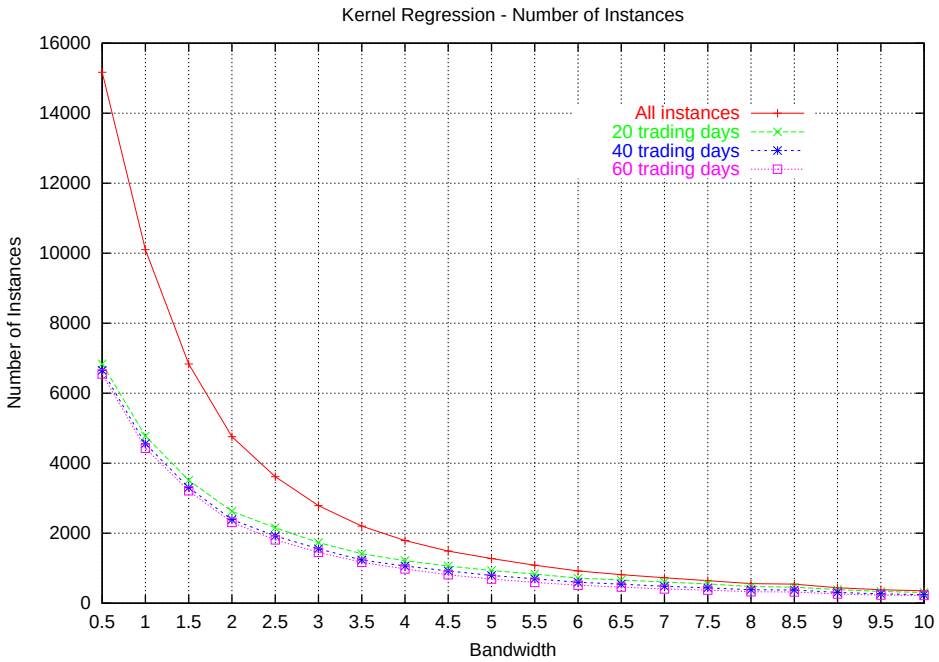

Figure 6.4.2c. The number of the "head-and-shoulder" instances when smoothing with Kernel Regression.

- The length of the pattern instances is linearly increasing with the increasing of the bandwidth. The mean lenght of the pattern instances is 126.

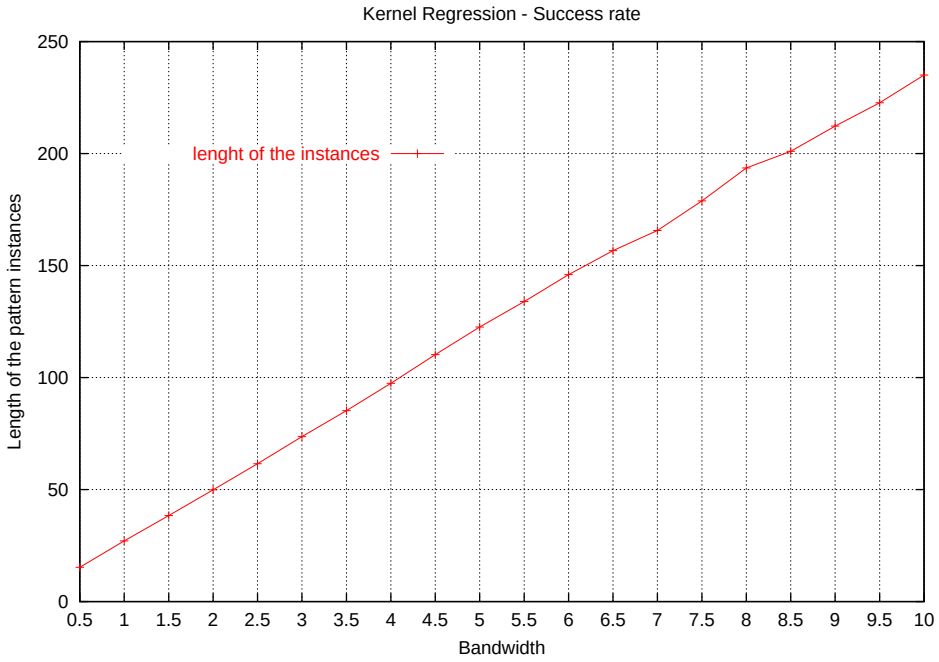

Figure 6.4.2d. The number of the "head-and-shoulder" instances when smoothing with Kernel Regression.

- The success rate is ranging between 40% and 70% for 40 and 60 trading days and between 45% and 98% for 20 trading days.

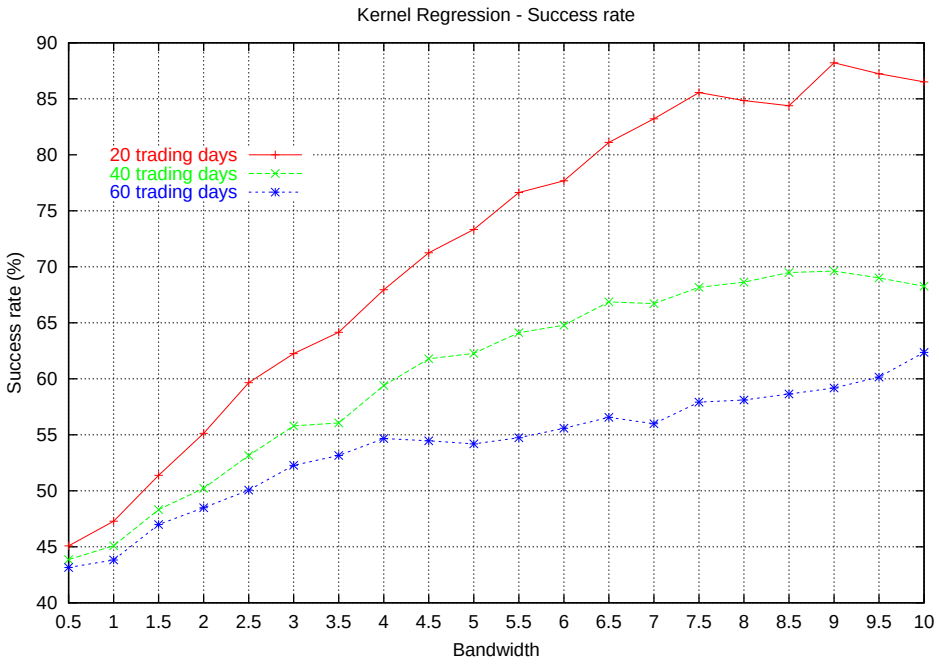

Figure 6.4.2e. The success rate with Kernel Regression.

6.4.3 Putting All Together

In this section we make a comparison between MDPP and Kernel Regression smoothing techniques based on the obtained results.

- Maybe the most important aspect that we discovered is the *best choice of the smoothing parameters*. The quality of the result depends directly on them because smoothing too much of the time series will also result in smoothing out some important peaks and bottom, and will lead to a lost of information. Selecting the smoothing parameters is an important challenge and requires more rigorous procedures. Some times, the optimal is not good for a particular case [15]. So, we find a lack of clear intuition for making the choice of the bandwidth, h.

  On the other hand, smoothing parameters used in the MDPP algorithm have intuitive meaning: D is the minimum number of days between every buying and selling and P is the minimum percentage of price movement stock traders consider to be significant. As simple as that. The answer to "how much can we smooth the data?" is now very clear and intuitive. Data smoothing can be adjusted very easily with MDPP smoothing, by simply choosing the right parameters, accordingly with our objectives.

- Although kernel regression is useful for its simplicity and intuitive appeal, kernel estimators suffer from a number of well-known deficiencies, for instance, *boundary bias*, *lack of local variability* in the degee of smoothing, and so on. The most obvious is the boundary bias. The regions around the X_t values at the boundary of a time series is asymmetric and thus the kernel estimate may be considerably biased. The main cause of boundary bias is that kernel regression does not differentiate between regions that have no values (i.e., the beginning and the end of the time series). It actually assumes that the kernel function goes to zero in areas outside the time series domain. Consequently, there is a strong bias towards zero in regions that are within a bandwidth of the boundary of the domain.

  A major weakness encountered in kernel estimators is the *curvature effects*, often referred to as "trimming the hills and filling the valleys". In essence, it means missing the important peaks and bottoms. The smoothness of a curve is determined by the rate of direction changes, not by the degree of absence of peaks and bottoms. Therefore, it is not required to remove major peaks and bottoms when smoothing a curve. In fact, peaks and bottoms are usually very significant, and are meaningful, especially in our context of stock prices. Treating them as insignificant noises and smoothing them away can result in a major loss of information and trade signals. The curves between extrema points are rather insignificant to the optimal trading strategies.

  In contrast, the MDPP algorithm has none of these problems. It keeps all the important peaks and valleys by adjusting the smoothing parameters. This gives not only an intuitively parameterized smoothing method but also a free-lagged result. Furthermore, MDPP is based on the "Landmark Model" which is intuitive, consistent with human intuition and episodic memory.

- Looking at the obtained results, both MDPP and Kernel Regression indicate that the "head-and-shoulder" pattern is indeed successful in predicting negative returns. However, comparing MDPP and Kernel, there are significant variations in the test results yielded.

  First, we have to assess that both of them reject the H_0 hypothesis with 95% confidence interval.

One obvious difference is that the total number of "good" pattern instances found using MDPP is generally larger than those found using Kernel regression. Another interesting aspect with respect to the number of instances is that in the MDPP case, for distance greater than 3 and a small or medium percentage, the CPL is cutting of almost a constant number of "bad" instances from the total amount of the instances found. On the other hand, Kernel finds a smaller number of instances which, for values of the bandwidth between 6 and 10, tend to be all "good" instances. For a greater percentage, the MDPP behaves somehow like the Kernel regression, and the number of the instances for distance greater than 16, is low.

The length of the instances found using MDPP is on average around 80 days for small percentage and 160 days for high percentage, so for a medium percentage, the average length will be around 120 days. Moreover, the average pattern length found using Kernel Regression is around 120 days.

From Bulkowski results, we know that the average pattern length should be around 3 months so, our results are consistent with his findings [2].

Let us concentrate further on the success rate of the pattern instances found.

For MDPP smoothing algorithm the results are very encouraging. For a small percentage, i.e P = 0.05, the success rate for 20 trading days is ranging from 46% to 62% with an average of 61%, the success rate for 40 trading days is ranging from 44% to 56% with an average of 55% and finally, the succes rate for 60 trading days is ranging from 43% to 52% with an average of 51%. When choosing a high percentage, i.e. P = 0.5, the success rate for 20 trading days is ranging from 45% to 100%, with an average of 81%, for 40 trading days is ranging from 44% to 100%, with an average of 80% and for 60 trading days is ranging from 43% to 100%, with an average of 70%.

When using Kernel Regression, the success rate for 20 trading days is ranging from 45% to 88%, with an average of 72%, for 40 days is ranging from 44% to 70%, with an average of 61% and for 60 trading days is ranging from 43% to 62%, with an average of 54%.

These results show that the MDPP algorithm not only finds much more instances than Kernel regression but also finds much more "good" instances, resulting in a higher success rate.

Most importantly is the comparison between the mean excess returns of both smoothing techniques. Figure 6.4.1a shows clearly that the MDPP algorithm outperforms Kernel Regression method. For 40 trading days, the mean average return for Kernel Regression is -3.628 which is less in absolute value than -4.149 which is the mean average return for MDPP with percentage greater than 0.1 and it is less in absolute value than -5.01 which is the mean return for MDPP with percentage of 0.1%.

- Kernel regression has the problem of incurring high computational cost for calculating all the distances and weights from the rest of the X_t values to the estimation point. In other words, in order to calculate the smoothed value at a particular time t, there is a need to compute the summation of the distances and weights over the rest of the X_t values. Thus, kernel regression is expensive in terms of computational complexity as it depends directly on the amount of data.

  On the other hand, MDPP algorithm is running in O(n) time, where n is the length of the time series.

6.5 Real Time Forecasting

In this section, we want to underline two important aspects. First, MDPP smoothing is not entirely applicable in real time forecasting. The reason is that, in real time, we are able to identify peaks or troughs only after they occur. So, we cannot use our MDPP smoothing technique for the trading strategies that are selling and buying at peaks and troughs.

On the other hand, trading a "head-and-shoulder" formation is not done at the peaks or troughs but instead at the neckline level and over a period of 20, 40 or 60 days. In this situation, head-and-shoulder is a particular case for which we can successfully apply MDPP smoothing for the purpose of finding "significant" landmarks at realtime.

7 Conclusions and Future Work

Forecasting future trend of the stock market is a long history of research. Probably the attractiveness of the end result has drawn interest from multiple disciplines of science to this problem.

In this paper we assess the predictive power of the "head-and-shoulder" pattern by comparing two smoothing techniques, MDPP and non-parametric Kernel Regression. Both MDPP and Kernel Regression indicate that the "head-and-shoulder" chart pattern is indeed successful in predicting negative excess returns. However, MDPP outperforms Kernel in almost all the tests that we have done.

The definition and the search of the patterns is done in our CPL language. Some of the salient features of the language are:

- It provides a high-level platform upon which analysts can define and search patterns easily without much programming expertise; this is different from work in neural networks which treat the pattern discovery process as a black box, something that the user can be uncomfortable with.
- Its specification is intuitive to how analysts would describe the patterns in human terms(fuzzy constraints), but without compromising on precision.
- Patterns are defined in a compositional style, which is a reflection of how the real analysts prefer to define.
- Allow an easy way to specify constraints and reuse them.

The research presented here has some very clear limitations that deserve to be addressed in future. First, it applies only to the Standard & Poor ($S\&P$) 500 stocks. Second, it considers only one market signal (the close price), even though technical analysis frequently asserts the importance of checking volume signals. Third, it covers a period of 10 years.

Since the "head-and-shoulder" pattern is considered by practitioners to be one of the most reliable of all chart patterns, it represent a natural point of departure for empirical research. Because

trading based on this pattern generates excess profits, investigating of other patterns may also be of interest.

References

[1] Thomas N. Bulkowski. The Head and Shoulders Formation. *Technical Analysis of Stocks and Commodities*, VOL 15:366–372, 1997.

[2] Thomas N. Bulkowski. *Encyclopedia of Chart Patterns.* John Wiley and Sons: New York, 2000.

[3] Chris Chatfield. *Time Series Forecasting.* Chapman and Hall CRC, 2001.

[4] Hardy C. Colburn. *The Investor's Guide to Technical Anaysis.* New York: McGraw Hill, 1978.

[5] Perng C.S., Wang W., Zhang S.R., and Parker D.S. Landmarks: A New Model for Similarity-Based Pattern Querying in Time Series Databases. *In Proceedings of 16th International Conferencing on Data Engineering (San Diego, California, IEEE Computer Society)*, pages 33–42, 2000.

[6] Beymer Davin and Tomaso Paggio. Image Representation for Visual Learning. *Science*, Vol 272:1905–1909, 1996.

[7] Nadaraya E.A. On estimating regression. *Theory of Probability Applied*, pages 186–190, 1964.

[8] Robert D. Edwards and John Magee. Technical Anaysis of Stock Trends, 6'th edition, jmibm. 1992.

[9] Savin G. and Weller P. Zvingelis J. The Predictive Power of "Head-and-Shoulders" Price Patterns in U.S. Stock Market. University of Iowa, USA. 2003.

[10] Watson G.S. Smooth regression analysis. *Sankhya*, Ser. A, Vol 32:359–372, 1964.

[11] Murphy J.J. *Technical Analysis of the Futures Market: A Comprehensive Guide to Trading Methods and Applications.* New York: Prentice Hall, 1986.

[12] Eamonn J. Keogh, Selina Chu, Davin Hart, and Michael J. Pazzani. An online algorithm for segmenting time series. *In Proceeding of the IEEE International Conference on Data Mining*, pages 289–296, 2001.

[13] Eamonn J. Keogh and Michael J. Pazzani. An enhanced representation of time series which allows fast accurate classificationm clustering and relevance feedback. *In Proceeding of the fourth ACM International Conference on Knowledge Discovery and Data Mining*, pages 239–243, 1998.

[14] Osler Carol L. and P.H. Kevin Chang. Methodical Madness: Technical Analysis and the Irrationality of Exchange Rate Forecasts. *Economic Journal*, VOL 109:636–652, 1999.

[15] Andrew W. Lo, Harry Manaysky, and Jiang Wang. Foundations of Technical Analysis: Computational Algorithms, Statistical Inference, and Empirical Implementation. *The Journal of Finance*, VOL 55:1705–1765, 2000.

[16] Anand S., Chin W.N., and Khoo S.C. Charting Patterns on Price History. *International Conference on Functional Programming (Florence, Italy, September 3 - September5*, pages 135–145, 2001.

[17] Armstrong J. S and Collopy F. Integration of statistical methods and judgement for time series forecasting: Principles from empirical research. *Forecasting with Judgement*, pages 269–293, 1998.

[18] Sameer Singh. Noise Impact on Time-Series Forecasting using an Intelligent Pattern Matching Technique. *Pattern Recognition*, VOL 32:1389–1398, 1999.

[19] Hardle Wolfgang. *Applied nonparametric regression.* Cambridge University Press, 1990.