Trading Strategies Based on K-means Clustering and Regression Models

Hongxing He[1], Jie Chen[1], Huidong Jin[1], and Shu-Heng Chen[2]

[1] CSIRO Mathematical and Information Sciences
GPO Box 664, Canberra, ACT 2601, Australia
Hongxing.He,Jie.Chen,Warren.Jin@csiro.au
[2] AI-ECON Research Center, Department of Economics
National Chengchi University, Taipei, Taiwan 11623
chchen@nccu.edu.tw

This paper outlines a data mining approach to the analysis and prediction of the trend of stock prices. The approach consists of three steps, namely, partitioning, analysis and prediction. A commonly used k-means clustering algorithm is used to partition stock price time series data. After data partition, linear regression is used to analyse the trend within each cluster. The results of the linear regression are then used for trend prediction for windowed time series data. Using our trend prediction methodology, we propose a trading strategy TTP (Trading based on Trend Prediction). Some results of applying TTP to stock trading are reported. The trading performance is compared with some practical trading strategies and other machine learning methods. Given the volatility nature of stock prices the methodology achieved limited success for a few countries and time periods. Further analysis of the results may lead to further improvement in the methodology. Although the proposed approach is designed for stock trading, it can be applied to the trend analysis of any time series, such as the time series of economic indicators.

Key words: Data Mining, Clustering, k-means, Time Series, Stock Trading

1 Introduction

Trend analysis and prediction play a vital role in practical stock trading. Experienced stock traders can often predict the future trend of a stock's price based on their observations of the performance of the stock in the past. An early sign of a familiar pattern may alert a domain expert to what is likely to happen in the near future. They can then formulate their trading strategy accordingly. Can we gain such knowledge automatically using data mining techniques ([6])? There have been studies on efficiently locating previously known patterns in time series databases ([1, 11, 3]). The

search for and matching of similar patterns have been studied extensively in time series analysis ([5, 9]). Patterns in long time series data repeat themselves due to seasonality or other unknown underlying reasons. Early detection of patterns similar to those that have occurred in the past can readily provide information on what will follow. This information will be able to help decision-making in regard to the trading strategy in stock market trading practice.

In this paper, we report an approach to the pattern matching of stock market time series data and apply it to stock trading practice. We use ten years of historical data to form training data. The k-means clustering algorithm is then applied to the training data to automatically create partitions. k clusters or representative patterns are formed. Each of these is presented as a time series, approximated by its cluster center. Each time series is then divided into two parts. The first part is used for pattern matching while the second part is used to decide the trend of the cluster. Binary classes in terms of trend are assigned to clusters. We use linear regression modeling to decide the classification of the representative patterns. The "UP" class is assigned to clusters with a positive gradient and the "DOWN" class is assigned to clusters with a negative or zero gradient. For each test time series, we look for the best match in all clusters. The trend class of the matched cluster is regarded as the future trend of the test series. Finally, we use the predicted trend of the test time series in our trading strategy to determine the trading decisions.

The paper is organised as follows. Section 2 explains the method for forming windowed time series for training and test data. Section 3 describes our methodology applied in trend analysis and its application to trading data matching and trend prediction. Section 4 explains two trading strategies used in our experiments. Section 5 gives the experimental designs used to run the simulation of this study. Section 6 shows some results and makes a comparison with some other trading strategies. Section 7 concludes the paper and lists possible future directions.

2 Time Series Data Preparation

We follow [3] closely to create training data by sliding a fixed-length time window from time t_b to t_e. The following $N = t_e - t_b$ time series are created with window length w_{tr}.

$$\begin{aligned} s_1 &: p_1, p_2, ..., p_{w_{\text{tr}}} \\ s_2 &: p_2, p_3, ..., p_{w_{\text{tr}}+1} \\ &\cdots \\ s_N &: p_N, p_{N+1}, ..., p_{w_{\text{tr}}+N-1} \end{aligned}$$

where $p_i (i = 1, 2, \cdots, w_{\text{tr}} + N - 1)$ are stock prices at time i. We therefore create an *N by* w_{tr} matrix or a data set with *N* data records and w_{tr} attributes. Note that all attributes take continuous values and conventional data mining methods can be applied directly.

We divide the windowed time series into two parts, the antecedent part and the consequent part (see Fig. 1). The antecedent data of length $w_{\text{te}} = w_{\text{tr}} - w_{\text{lm}}$ is used

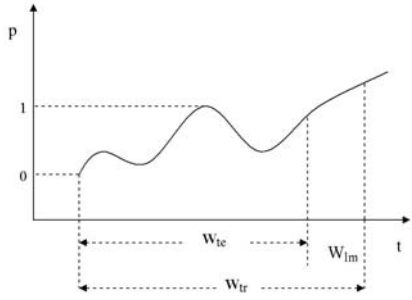

Fig. 1. Schematic view of windowed time series and normalisation

to decide the cluster index. The consequent part of length $w_{lm} = w_{tr} - w_{te}$ is used to decide the classification of the cluster. In order to do the trend pattern matching, we normalise each windowed series individually. The windowed time series used for the clustering algorithm is normalised in such a way that the first w_{te} part of a series is normalised to fall between 0 and 1. The data is normalised in this way to ensure that the time series to be matched in order to decide the cluster index are on the same scale. Fig. 1 is a schematic view of a windowed time series.

3 Methodology for Trend Analysis

Our data mining approach consists of the following steps.

1. Initialisation.
   - Select window lengths w_{tr} and w_{te} for the antecedent and the consequent part of time series, respectively.
   - Select a training period.
     In this paper, we consider a ten-year training period. For example, a training sample starts w_{tr} days before the first trading day of the year 1991 and ends on the last trading day of the year 2000.
   - Select a test period.
     The test period is set to 2 years long. For example, in the above-mentioned example, the corresponding test period is 2001-2002. The test period then starts from the first trading day of 2001 to the last trading day of 2002.
2. Data Mining.
   - Create N training series of window length w_{tr} from the training period.
   - Normalise each time series individually such that the first w_{te} values of the time series fall between 0 and 1.
   - Partition the training data into k clusters, which are represented by their cluster centers.
   - Classify all the clusters into two distinct classes using a linear regression model. A model is built based on the last w_{lm} values of each cluster center. Class “UP” is labeled if the gradient is positive and “DOWN” otherwise.

3. Test model on test data.
   - Form a test series dataset with the window length w_{te}. Normalise them individually. Consequently, values will fall between 0 and 1.
   - Assign a cluster label $c_i = j$ to time series s_i in the test data such that cluster $j(j = 1, 2, \cdots, k)$ has the smallest Euclidean distance to the normalised series s_i.
   - Assign the class ("UP" or "DOWN") of cluster j to time series s_i, where time series s_i has cluster label j.
   - Calculate returns for a selected trading strategy.

Some details are given in the following subsections.

3.1 Unsupervised Learning Using *k*-means Clustering

k-means clustering is an algorithm to group objects based on attributes or features into k groups ([10]). $k > 1$ is a positive integer. The input of the k-means algorithm is the training data set with N records each with w_{tr} attributes (in our case). The grouping is done by minimising the cost function defined in (1), that is, the sum of distances between data and their corresponding cluster centres.

$$\text{Cost} = \sum_{j=1}^{k} \sum_{i=1}^{N} D(s_i, \text{Centre}(j)) \delta(c_i, j) \tag{1}$$

The Kronecker delta function $\delta(c_i, j)$ means that only members of cluster j are included in the sum. Thus, the purpose of k-means clustering is to partition the data into a number of groups automatically in such a way that data records within the same cluster are similar and data records belonging to different clusters are dissimilar.

The k-means clustering procedure can be stated as follows:

Step 1: Initialise k cluster centres randomly. Assign all the data records in the training set to their nearest cluster centres.

Step 2: Update cluster centres using (2).

Step 3: Update cluster label assignment for all the data records in the training set.

Repeat steps 2 and 3 until convergence is achieved, i.e., until a pass through the training data causes no new assignments.

$$\text{Centre}(j, m) = \frac{\sum_{i=1}^{N} p_{i,m} \delta(c_i, j)}{\sum_{i=1}^{N} \delta(c_i, j)} \tag{2}$$

In (1) and (2), c_i is the cluster label for the ith record, N is the total number of records, $p_{i,m}, m = 1, 2, \cdots, w_{\text{tr}}$ is the mth attribute of the ith record and $j = 1, 2, \cdots, k$ stands for the jth cluster.

It is well known that k-means clustering is a greedy search for the minimum cost function [6, 8]. The local minimum rather than the global one is reached as a consequence. In order to obtain a good data partition, we run a number of independent training procedures (each with a different random number). The resultant clustering with the lowest cost out of these runs is used.

3.2 Classification of Clusters

As stated earlier, we use a linear regression model to determine the trends of clusters. Extracting the trend from the slope of the model is straightforward. This model is also more robust than other complex models.

In linear regression [2], a dependent variable is fitted by a straight line expressed by (3). The constants a and b are determined by minimising the mean square error. In our case, the constants a and b are given by (4) and (5), respectively.

$$y = a + bt \tag{3}$$

$$a = \frac{\sum_{i=1}^{t_{\text{lm}}} y_i}{w_{\text{lm}}} - b\frac{\sum_{i=1}^{t_{\text{lm}}} t_i}{w_{\text{lm}}} \tag{4}$$

$$b = \frac{\sum_{i=1}^{t_{\text{lm}}} t_i y_i - \sum_{i=1}^{w_{\text{lm}}} t_i \sum_{i=1}^{t_{\text{lm}}} y_i / t_{\text{lm}}}{\sum_{i=1}^{t_{\text{lm}}} t_i^2 - (\sum_{i=1}^{t_{\text{lm}}} t_i)^2 / t_{\text{lm}}} \tag{5}$$

Trend class "UP" is assigned to clusters with $b > 0$, and "DOWN" to other clusters.

3.3 Pattern Matching

For a test time series $s_i = s_{i,1}, s_{i,2}, \cdots, s_{i,w_{te}}$, we assign a cluster label as given by (6)

$$cl(i) = \arg\min_j MSdis(Centre(j), s_i) \;\; j = 1, 2, \cdots, k \tag{6}$$

where $cl(i)$ is the cluster label of test time series s_i. The mean square distance between time series s_i and the cluster centre of j is expressed by (7).

$$\text{MSdis}(\text{Centre}(j), s_i) = \frac{\sum_{m=1}^{w_{\text{te}}} (\text{Centre}_m(j) - s_m(i))^2}{w_{\text{te}}} \tag{7}$$

Since the test time series and cluster centre time series are both normalised to (0,1), the distance indicates their difference in shape. The test series is then assigned a trend class ("UP" or "DOWN") by simply assigning the trend class of the cluster closest to it.

4 Trading Strategies

In this section we introduce two trading strategies. The first strategy is naive trading, where the future trend is not taken into consideration. The second is the same as the first except that the future trend prediction is used in the trading decision.

If π_t = 1 :
    if $p(t)(1-c) > p(t_{\text{prev\_buy}})(1+c)$
        Action = Sell
    else :
        Action = Stay
If π_t = -1 :
    if $p(t)(1+c) < p(t_{\text{prev\_sell}})(1-c)$
        Action = Buy
    else :
        Action = Stay

Fig. 2. Naive Trading (NT)

4.1 Naive Trading (NT)

We call our trading strategy "naive trading" because it is very simple. In NT, we buy the stock if we are not holding a share and the purchase cost is lower than the value at which we sold previously. By the same token, we sell the stock if we hold a share and we can make profits from that sale. That is, we sell the stock if the value received exceeds the value at which we bought previously. The trading strategy is expressed in Fig. 2.

In Fig. 2, c is the trading cost. π_t is the state of the stock holding at time t ($\pi_t = 1$ for holding, -1 for not holding). $p(t_{\text{prev\_buy}})(1+c)$ is the cost of buying in previous buying action. $p(t_{\text{prev\_sell}})(1-c)$ is the value received in the previous selling action. We initialise the trading on the first day of the time period by buying a unit of stock.

4.2 Trading based on Trend Prediction (TTP)

As illustrated in Fig. 3, TTP is a slight variation of NT. The only difference is that we consider the forward trend of the stock price. Intuitively, we would buy a stock when the "UP" trend is found and sell it when the "DOWN" trend will follow. Since we try to enter the market as soon as we can, we only use the trend information to select the time of selling. We sell the share only if the trend prediction is downward. In other words, instead of selling a stock at any profit, we hold it until a maximum profit is reached.

5 Experimental Design

5.1 Training and Test Periods

In order to compare our trading strategies with the strategies studied in [4], we use five stock indexes which were used in [4] with the same trading periods and testing periods as indicated in Table 1.[1]

[1] Reference [4] also shows the results of the 21 technical trading strategies, which have actually been used by investors in financial securities firms. These strategies are basically

If π_t = 1 :
    if $p(t)(1-c) > p(t_{\text{prev\_buy}})(1+c)$ and $Trend(t)$ ="DOWN"
        Action = Sell
    else :
        Action = Stay
If π_t = -1 :
    if $p(t)(1+c) < p(t_{\text{prev\_sell}})(1-c)$
        Action = Buy
    else :
        Action = Stay

Fig. 3. Trading based on Trend Prediction (TTP)

Table 1. List of Three Test Time Periods and their Corresponding Training Periods

	Test Period	Training Period
1	1999–2000	1989–1998
2	2001–2002	1991–2000
3	2003–2004	1993–2002

5.2 Parameter Values

The values of the parameters used by k-means clustering are summarized in Table 2. The performance and efficiency of the algorithm can depend on the number of clusters and the number of runs. While low values of these two parameters may make the derived results less robust, high values of them can cause the computation to be very time-consuming. The values chosen here are the results from a few pilot experiments. The lengths of the windowed time series and its constituents (the antecedent part and the consequent part) are decided on pilot experiments as well. The decision on the length of the antecedent part of the windowed series can determine the patterns to be recognized, whereas the length of the consequent part can affect the statistical errors of the regression model.

6 Experimental Results

6.1 Accuracy Rate

The essential idea in this paper is whether we can meaningfully cluster the windowed financial time series based on their associated geometrical patterns. The clustering

made up of the historical data on prices and trading volumes. Since the data on trading volumes are not available from some markets, their testing is inevitably limited to those markets whose data are sufficient, which includes the US, UK, Canada, Taiwan and Singapore. This is the main reason why we only consider the stocks of these five countries in this study. Nonetheless, in the case of Japan, since the trading volume data are available for the period 2003-2004, the results for Japan for that period also become available. We therefore consider six countries, instead of five, in the period 2003-2004.

Table 2. List of Parameters and their Values

Name	Description	Value
k	Number of clusters	50
N_{run}	Number of k-mean runs	20
w_{tr}	Length of each windowed series	50
w_{te}	Antecedent part of each windowed series	40
c	Transaction cost rate	0.005

Table 3. The Accuracy Rate of the Trend Prediction

	US	UK	Canada	Taiwan	Singapore	Japan	Mean
1999-2000	52.1	53.0	53.9	52.9	52.0	NA	52.8
2001-2002	50.2	48.6	51.7	49.9	49.1	NA	50.2
2003-2004	56.3	51.1	50.0	52.4	51.7	52.4	52.3

work can be useful if it helps us predict the future from the past. In other words, we can perform better conditional expectations when provided with the geometric patterns. Our design of the TTP trading algorithm is in effect based on this assumption. Therefore, it would be important to ask whether clustering does help predict the trend.

In Table 3, we report the accuracy rate achieved by our trend-prediction algorithm, i.e., the prediction based on the sign of the regression coefficient (see Eqs. 3–5). The overall accuracy rate is 51.8% over about a total of 7,500 predictions. While the 1.8% margin over the accuracy rate of random guessing, i.e., 50.0%, is small, it is statistically significant at a significance level of 0.05.

Furthermore, if we look at countries by countries and periods by periods, we find that out of the 16 scenarios, there are only three scenarios (the UK, Taiwan and Singapore), all of them happening in the period 2001-2002, for which the accuracy rate is below 50%. For the rest, the accuracy rates are all above 50%. Therefore, our proposed trend prediction algorithm based on k-means clustering does bring us some prediction power, although it is somewhat marginal. We shall come back to this point again in the last section.

6.2 Results of the Total Return

Given the little prediction power, it would be interesting to see whether we can take advantage of it. Tables 4 to 6 list the returns from using the TTP strategy for selected markets in the three time periods. In addition, we also compare the performance of our trading strategies with the NT strategy and those studied in [4], which include the Buy-and-Hold (B&H) strategy and the trading strategies developed by machine intelligence, specifically, genetic programming (GP).

In Tables 4 to 6. The row headed by B&H is the total return earned by simply using the B&H strategy, the row headed by GP1 refers to the mean total return of the 50 GP runs without short sales, whereas the row headed by GP2 refers to the

Table 4. The Total Return of Stock Trading for 1999–2000

Rule	US	UK	Canada	Taiwan	Singapore
B&H	0.0636	0.0478	0.3495	-0.2366	0.3625
GP 1	0.0655	0.0459	0.3660	0.1620	0.1461
GP 2	0.0685	0.0444	0.3414	0.5265	0.1620
TTP	0.1778	0.1524	0.0541	-0.22	0.4654
NT	0.0786	0.1560	0.0207	-0.1480	0.0524

Table 5. The Total Return of Stock Trading for 2001–2002

Rule	US	UK	Canada	Taiwan	Singapore
B&H	-0.3212	-0.3682	-0.2395	-0.1091	-0.2998
GP 1	-0.3171	-0.3625	-0.1761	0.0376	-0.3123
GP 2	-0.3231	-0.3658	-0.2367	-0.0470	-0.2939
TTP	-0.3196	-0.3560	-0.2413	0.0327	-0.2636
NT	-0.3190	-0.3545	-0.2345	0.0511	-0.2636

same thing but with short sales.[2] The performances of B&H, GP1 and GP2 are all borrowed from [4], and are duplicated in Tables 4 to 6. The rows headed by TTP and NT give the mean total return by taking an average over the 20 runs.

Based on the results shown in these tables, we would like to first ask whether the TTP strategy outperforms the NT strategy. Notice that the NT strategy is a very conservative strategy: it basically does not recommend any transaction which may result in an immediately realized loss. Based on its "stubborn" rule without any stop-loss design, it can be easily locked in and can trigger a very risky decision. Furthermore, since it only cares about the occurrence of the current realized loss, and cares neither about future gains nor future losses, it does not learn anything from the past, and hence is not intelligent. Can the TTP strategy which has little power in predicting future trends beat this naive strategy?

Out of the 16 scenarios which we have examined, TTP loses six times to NT. While, from these numbers, TTP is still by and large dominant, NT does not perform in a way that is as inferior as one might expect. Even though in many cases the accuracy rate of trend prediction is over 50%, TTP can still give way to NT, which means that, not only in prediction, but also in profits, clustering only has a marginal contribution to the trading design.

The second thing that we would like to look at is to compare our proposed trading strategy with some of the other trading strategies developed using different computational intelligence tools, such as genetic programming. Genetic programming applies the ideas of biological evolution to a society of computer programs. From one generation to another, it follows the survival of the fittest principle to select from the existing programs, and then operates these selected programs via some genetic oper-

[2] The *total return* means the return earned over the entire investment horizon [0,T] per unit of investment. So, say, $R = 0.05$ means that a one-dollar investment will earn five cents as a total over the entire investment horizon [0,T]. For details, see [4].

Table 6. The Total Return of Stock Trading for 2003–2004

Rule	US	UK	Canada	Taiwan	Singapore	Japan
B& H	0.3199	0.1888	0.3625	0.3434	0.5311	0.3054
GP 1	0.3065	0.1797	0.3109	0.3631	0.2512	0.0212
GP 2	0.3109	0.1817	0.3389	0.2740	-0.0183	-0.1792
TTP	0.1461	0.0983	0.0049	0.2239	0.0351	0.0044
NT	0.0302	0.0100	0.0049	0.0324	0.0234	0.0025

ators, such as reproduction, crossover and mutation. Each program in GP can serve as a trading strategy.

The representation of trading strategies developed by GP is largely determined by the primitives, i.e., the function set and the terminal set. The standard choice of these primitives makes the resultant trading strategies more like rule-based trading strategies, which may or may not be associated with the geometry-based strategies. Therefore, the comparison between GP and k-means can provide us with some insight into different ways to think of trading strategies. Besides, some severe limitations of using GP have been observed in [4]; therefore, it would also be interesting to know whether when GP performs particularly badly there are other alternatives that can take its place. In the latter, we are actually inquiring into the cooperative relationship rather than just the competitive relationship between two intelligent machines.

Our results generally show that, out of 16 scenarios, GP1 wins 10 times, whereas TTP picks up the remaining six. In particular, in the period 2003-2004, GP1 consistently dominates TTP. Of course, one should always be careful when making comparisons between two different intelligent machines, since there are many different variants determined by different control parameters. Therefore, while the specific results seem to favor GP1 over k-means, one can hardly obtain a decisive result given the limited evidence and limited trials. In these experiments, there is no uniform dominance for either GP1 or k-means.

Finally, we also find that the performance of GP1 is also superior to NT.

7 Conclusions and Future Work

We have applied a data mining approach to analyze and predict the trend of stock prices and have developed trading strategies based on these predictions. The proposed trading strategy based on trend prediction using k-means clustering is applied to a few selected countries over some samples of financial time series. The results are compared with various existing trading strategies and GPs obtained earlier in [4]. Although the trend analysis method is simple and does not always achieve high accuracy, it has the following advantages:

1. It allows matching every testing time series to one of the representative patterns. Therefore, a decision regarding trading can always be made. This feature is not shared by other sophisticated search techniques, such as the motif, which can be

used to locate patterns in the history, but may not be able to match *every* testing time series, and hence is not always able to make a trading decision ([11]).
2. The computation resources required are an important issue in real world applications. We use a computer with an Intel(R) Xeon(TM) CPU (3.20 GHz) running the Linux operating system. The procedural language Python is used for programming. It takes about 40 minutes to finish training and testing on a ten-year training period followed by a two-year testing period. The huge saving in time makes it a competitive candidate in real world applications.

While the methodology developed in the work can correctly predict the trend of stock prices in some markets in some periods, it does not perform well on many occasions. The fundamental difficulty arises from the highly volatile nature of the stock price. The proposed trend-prediction algorithm is very simple, and, therefore, has its limitations. For the future, we propose the following directions to enhance its performance:

1. Improve the performance by optimizing the parameter settings. We may add a validation period to optimize the setting of various window lengths and the number of clusters.
2. Improve the linear regression model with a larger window length, and perhaps by allowing some overlaps between the antecedent part and the consequent part.
3. In this paper, the decision on the classification of clusters is made based on the sign of the regression coefficient. However, for those samples whose coefficient values are around zero, a slight difference can have a big effect on the final decision, be they up or down. We may consider using fuzzy mathematics or something equivalent to smooth this dichotomous decision so as to improve the accuracy of the trend prediction.
4. Improve computational efficiency by using sophisticated and scalable clustering techniques, such as [8, 7].
5. By introducing scale change to pattern matching we can discover similar patterns with different time scales.
6. In this paper, we use the trend prediction only for making selling decisions. With better and more accurate trend prediction, we may modify the trading strategy TTP by using the trend prediction for making buying decisions as well.
7. Combine our method with other techniques, such as GP, for better and more sophisticated trading strategies.

Acknowledgments

The authors are grateful to the authors of [4] for their permission to use their data and results in this paper. The authors also acknowledge Damien McAullay and Arun Vishwanath for their assistance in the preparation of the paper.

References

1. Agrawala R, Faloutsos C, Swami A (1993) Efficient similarity search in sequence databases. In: Proceedings of the 4th international conference on foundations of data organization and algorithms:13–15
2. Chambers JM, Hastie TJ (eds) (1992) Statistical models in s. Chapman & Hall/CRC
3. Chen SH, He H (2003) Searching financial patterns with self-organizing maps. In: Chen SH, Wang PP (eds) Computational intelligence in economics and finance. Springer-Verlag:203–216
4. Chen SH, Kuo TW, Hsu KM (2007) Genetic programming and financial trading: how much about "what we know"? In: Zopounidis C, Doumpos M, Pardalos PM (eds) Handbook of financial engineering. Springer. Forthcoming.
5. Ge X(1998) Pattern matching financial time series data. Project Report ICS 278, UC Irvine
6. Han J, Kamber M (2001) Data mining: concepts and techniques. Morgan Kaufmann Publishers, San Francisco, CA, USA
7. Jin HD, Leung KS, Wong ML, Xu ZB (2005) Scalable model-based cluster analysis using clustering features. Pattern Recognition 38(5):637–649
8. Jin H, Wong ML, Leung KS (2005) Scalable model-based clustering for large databases based on data summarization. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(11):1710–1719
9. Keogh E, Smyth P (1997) A probabilistic approach to fast pattern matching in time series databases. In: Proceedings of KDD'97:24–30
10. MacQueen JB (1967) Some methods for classification and analysis of multivariate observations. In: Proceedings of 5-th Berkeley symposium on mathematical statistics and probability. Berkeley, University of California:281–297
11. Patel P, Keogh E, Lin J, Lonardi S (2002) Mining motifs in massive time series databases. In: Proceedings of the 2002 IEEE international conference on data mining:370–377