Distance-Based High-Frequency Trading

Travis Felker[1], Vadim Mazalov[1,2*] and Stephen M. Watt[2]

[1] Quantica Trading, 119 King St. West Suite 300, Kitchener, Ontario, Canada N2G 1A7
travis@quanticatrading.com
[2] University of Western Ontario, London, Ontario, Canada N6G 2A5
Stephen.Watt@uwo.ca

Abstract

The present paper approaches high-frequency trading from a computational science perspective, presenting a pattern recognition model to predict price changes of stock market assets. The technique is based on the feature-weighted Euclidean distance to the centroid of a training cluster. A set of micro technical indicators, traditionally employed by professional scalpers, is used in this setting. We describe procedures for removal of outliers, normalization of feature points, computation of weights of features, and classification of test points. The complexity of computation at each quote received is proportional to the number of features. In addition, processing of indicators is parallelizable and, therefore, suitable in high-frequency domains. Experiments are presented for different prediction time intervals and confidence thresholds. Predictions made 10 to 2000 milliseconds before a price change resulted in an accuracy that ranged monotonically from 97% to 75%. Finally, we observed an empirical relation between Euclidean distance in the feature space and prediction accuracy.

1 Introduction

We take the problem of high-frequency trading to be short term prediction of the price of assets. A principal difficulty of research in this area is the considerable number of variables and complicated cause-effect relationships among these variables and potential outcomes. Some investigators therefore view the market as unpredictable, or following a "random walk", due to a large number of random factors [8]. Conversely, there are groups of researchers who argue that there is merit in developing predictive models for financial instruments.

In this paper, we consider the market to be an organized and coherent environment in which short term changes in stock prices occur due to certain actions of market participants and do not follow a random walk. Our objective is to model these changes. We aim to achieve short term forecast, as opposed to the long term, since the number of contributing factors is smaller, and their values are easier to measure. Also, the possibility of a negative impact by outside factors, e.g. an unscheduled news release, is lower and has less effect in the short-term. The difficulties increase with prediction interval and may quickly become intractable. This longer horizon, however, is of less immediate interest to us—the ability to predict a price change in a time interval sufficient to execute a transaction can be a valuable asset.

This problem can be seen as a one of pattern recognition:

- A significant amount of data is available from the market
- Some data is associated with specific events, e.g. asset price changes, and can be used in training.
- The model should be able to analyze incoming data based on the training data sets and make decisions in real time.

*Presently at Amazon Canada, 400-410 Adelaide St. West, Toronto, Ontario, Canada M5V 1S8.

Unlike many other applications of artificial intelligence, the pressures on high-frequency trading (HFT) software are typically severe. Two that are particularly applicable in the domain include: pressure for increased robustness (being able to operate correctly and reliably in different market conditions) and pressure for reduced latency. Success of a solution depends on differences of milliseconds and sometimes microseconds. Since execution time is typically reversely correlated with robustness, the underlying machine learning model must be very efficient. In addition, due to the magnitude of high-frequency data, it is not feasible to handle it by a statistical model in its raw form. Data mining and feature extraction become essential.

A number of trading strategies have been developed over time. These vary greatly from simple and fast, typically implemented by participants who have heavily invested in advanced network infrastructure, to more sophisticated models, placing more emphasis on execution logic and model robustness. Several techniques have been published derived from different mathematical models, e.g. from time series analysis [2, 5, 12], neural networks [1, 6, 13], support vector machines [11], hidden Markov models [9], and nearest neighbours [7, 10]. In general, to be able to apply a machine learning technique, one should first select features that describe the events to be modelled. In HFT, these features can be computed from a variety of *market events*, including the price change of an asset, dynamics of its bid and offer depths, trading volume, and similar features of complementary and supplementary assets across different exchanges. In some cases, technical indicators are selected and optimized by genetic algorithms [3].

Our HFT paradigm is centered around accurate and highly efficient short-term prediction of *one* change in the price of an asset. Rather than hard-coding *ad hoc* trading strategies, our method monitors the market and sets up a space of observations, then measures how close the real-time data is to the recorded observations. Predictions are based on numerical indicators computed from data received from the market. Indicators can be computed in parallel and therefore decisions can be delivered quickly. In an attempt to make the recognition algorithm as efficient as possible, we deploy perhaps the simplest form of classifier — one using the feature-weighted Euclidean distance to the centroid of a corresponding training cluster.

The model uses two clusters of points in the space of technical indicators. One cluster represents a price movement up and the other represents a price movement down. Hypothetically, the centroid represents the combination of perfect values of indicators at the time of an event. The weight of an indicator is computed based on the standard deviation of the features in the cluster. When the number of points in a training cluster reaches a predefined threshold, the cluster is processed as follows: First, the outliers are removed. Then the centroid of the cluster is computed. Finally, the coordinates of the centroid are normalized.

Once both clusters have been processed at least one time, the system starts to classify new feature points computed with every market event and makes a prediction when the classification confidence is high enough. Groups of new points are introduced to the training clusters, keeping the clusters updated and making the model adaptable. To our knowledge, there is no published work detailing this model and the related technical indicators.

The remainder of this article is organized as follows. In Section 2, we present the necessary equity market basics and terminology, and the concept of high-frequency trading as applicable to this paper. Section 3 outlines the set of technical indicators and the procedures of removal of outliers, normalization and computation of weights. Section 4 describes the classifier. The complexity of the model is discussed in Section 5. The experimental setting and results are presented in Section 6. Section 7 concludes the article and suggests future work.

2 Preliminaries

2.1 The Stock Market

Unlike many securities and derivatives exchanges, US equity markets — such as the New York Stock Exchange (NYSE), the National Association of Securities Dealers Automated Quotations (NASDAQ), or the American Stock Exchange (AMEX) — offer several venues where the same product may be bought and sold. Thus a market for a particular product (or asset) is comprised of several quotes at each instant. These markets, known as Alternative Trading Systems (ATSs), represent challenges associated with the number of data sources and market asymmetries.

Having multiple venues causes inherent inefficiencies which may also be viewed as opportunities. One such opportunity, used in this experimental model, is the idea that bids and offers at best available price (known as the National Best Bid/Best Offer, abbreviated NBBO) will be bought and sold asynchronously, creating a scenario where shares are available on one venue while no longer available at the NBBO on another. The order in which the venues are bought or sold is largely dictated by the costs involved in removing liquidity, as well as the available liquidity on any particular venue. When an exchange or ATS is no longer available at the NBBO it can be said to have *left* the bid/offer. A price change, in this model, is defined as all available exchanges leaving a product's NBBO on one side. A price change up is signified by all available offers leaving an NBBO and a price change down is signified by all available bids leaving an NBBO.

An order entering one ATS may have permission to be routed to another ATS or may be restricted to be executed locally. It may be the case that an order to buy (*bid*) at the best offer is directed to an ATS which does not have an offer to sell (*ask*) at that price. If such an offer has instructions to not route away, it cannot interact with the orders at that price. When this occurs the best bid becomes equal to the best offer and the market becomes *locked*. The same is true with respect to an offer to sell that occurs at the best bid.

Many data packages offer quotes for a *composite* exchange. In all cases, this quote refers to the venue which is displaying the greatest liquidity at a specific price. One composite quote is displayed at each price and as such the composite exchange can represent different exchanges/ATS at different prices. In the case that a new bid or offer is added to a venue at a price which causes it to become the venue with the largest advertised volume, the composite will then become representative of this venue. Several analytics considered in this paper examine the composite quote.

2.2 High-Frequency Trading

High-frequency trading is defined as any consistent trading activity with a significantly brief time span, and a high number of daily discrete round turns (completed trades) and messages. While our model does not exclusively apply to high-frequency trading, the effectiveness of the model is directly correlated with the chronological proximity to events. Therefore, the model will be greatly affected by factors that normally affect high-frequency trading. These factors include: infrastructure and latency, clearing fee structure, and software optimization.

Let t be the time interval it takes to send a buy/sell signal to an exchange/ATS and have it executed. If a correct prediction has occurred and the price change happens earlier than a time t after the prediction, the buy/sell signal will not be matched at the exchange and the order can be cancelled without any losses. If a prediction is incorrect, independently of the time of the price change, some losses are likely to occur. Since trades governed by this model are not designed to capture large price movements, the fees paid to an exchange, clearing house, and applicable taxes become important factors to consider. These transaction fees can lessen gains from successful predictions and significantly increase loses from unsuccessful predictions.

3 Technical Indicators

3.1 The Set of Indicators

The number of possible technical indicators that can be extracted from the stock market is overwhelming. Counter-intuitively, including all seemingly relevant indicators does not necessarily yield the highest performance. For a fixed training set, as the dimensionality of the feature vector increases, the sample points become more sparse and the performance will eventually start to decrease. This has been called the ”peaking phenomenon” [5]. Thus, careful selection of the indicators is important. They should not be redundant and should fully describe the asset at the time of an event.

In an effort to limit the available indicators to a manageable number and correctly predict a price change, we examine only the quotes at the current best bid and ask. We consider indicators that describe the activity of a single product independently of complementary and supplementary securities. Features are divided into those that are common for all exchanges considered and those that are relevant only to the composite.

To demonstrate classification under the presented model, we investigate only the simplest indicators. The ones computed for every exchange are described below

- Weighted rate of change (*ROC*) of the relation of the bid depth (number of shares bid) to the offer depth (number of shares offered)

  $$ROC_i = w\frac{b_i/o_i - b_{i-1}/o_{i-1}}{t_i - t_{i-1}} + (1-w)\, ROC_{i-1}$$

  where b_i and o_i are respectively the bid depth and the offer depth of a quote i, t_i is the time when the quote was received, $t_i - t_{i-1}$ is the time interval between the quotes, and w is a weight.

- n_{cb} (n_{co}) – the number of times an exchange/ATS locked the market on the bid (offer).

- n_{lb} (n_{lo}) – the number of times an exchange/ATS left the NBBO on the bid (offer).

Within every exchange, each of the indicators is assigned a weight. The weight of an exchange is determined as the average weight of its indicators. Then we compute the composite indicators:

- s_b (s_o) – the sum of weights of exchanges/ATSs whose bid (offer) price is equal to the NBBO bid (offer).

The set of features forms a D-dimensional point, where $D = 5E + 2$ and E is the number of exchanges considered. All of the features are calculated from data received after the previous change in price. Therefore, the market behaviour before the previous change does not have an effect on the point evolving from the incoming data.

3.2 Outliers

We use each market event in the computation of features. Most events do not represent a price change and are used only to compute distances, i.e. in classification. However, the feature vector computed from the event that indicates a change in price is added to the corresponding training cluster, unless it is considered an outlier.

Outliers represent unusual market behaviour, and the feature points from these non-stable market should not be used in training. We assume that the values of indicators are normally distributed. The method we use to determine if a point is an outlier is based on the three-sigma rule as shown in Algorithm 1. Essentially, the procedure is run on the cluster periodically. We chose to run the procedure whenever the cardinality of the cluster is divisible by 5.

Algorithm 1 IsOutlier(x)

Input: x, a D-dimensional feature point to be tested whether it is an outlier.
Output: *true* if the point is an outlier, and *false* otherwise.

```
for $i = 1$ to $D$ do
  if $|x_i - c_i| > 3\sigma_i$ then
    {where $x_i$ is the value of the $i$-th feature of the point $x$, $c_i$ is the average value of the feature among all points in the cluster (the $i$-th feature of the centroid of the cluster), and $\sigma_i$ is the standard deviation of the feature among all points in the cluster.}
    return true
  end if
end for
return false
```

3.3 Normalization

After the removal of outliers, the values of indicators are normalized. We are using isotropic distance measures in the feature space to determine close observations. This implies that closeness in different directions must be comparable. Since the features are fundamentally different in nature and can vary significantly in their absolute values, normalization is necessary.

To achieve this, it is usually necessary to perform some numerical transformation on the feature values. The transformations should map feature values to the same range (we choose the real interval $[0, 1]$) and have the same condition number. Informally, this means that, after transformation, equal numerical changes in two features should have the same importance. In principle, the choice of transformation depends on the phenomenon observed and a variety of forms may be applicable, e.g. power laws, exponential transformations, sigmoidal or trigonometric transformations, and so on. In practice, these transformations are often built into the definition of the features themselves, and we assume this is the case in the remainder of this paper. To perform the normalization, we find the maximum x_i^M and the minimum x_i^m values of an indicator i among all points in the cluster, and then normalize the feature as

$$x_i' = \frac{x_i - x_i^m}{x_i^M - x_i^m}.$$

These steps are described in more detail in Algorithm 2. Denormalization of a point is the inverse. We note that the outlier test described in Algorithm 1 is invariant with respect to normalization: An outlier can be detected whether its original values are tested against the original values of training samples, or whether it is normalized first (with respect to the maximum and minimum values in the cluster) and tested against the normalized values.

For a point p to be an outlier, at least one of its coordinates p_o should satisfy $|p_o - s_o| > 3\sigma_o$. The case $p_o - s_o > 3\sigma_o$ implies

$$p_o > \frac{1}{K}\sum_{i=1}^{K} x_{oi} + 3\sqrt{\frac{1}{K}\sum_{i=1}^{K}(x_{oi} - \frac{1}{K}\sum_{j=1}^{K} x_{oj})^2}, \tag{1}$$

where K is the number of points in a cluster and x_{ij} is the i-th feature of the j-th point. Applying

Algorithm 2 Normalize(x)

```
Input: $x$,a $D$-dimensional feature point to be normalized.
Output: $x'$ the normalized point.
  {Find the maximum and the minimum values of each indicator among the points in the cluster.
   In practice, this is done only once for the set of points.}
  for $i = 1$ to $D$ do
    {$K$ is the number of points in a cluster and $x_{ij}$ is the $i$-th feature of the $j$-th point}
    $x_i^M \leftarrow \max_{j=1..K} x_{ij};\quad x_i^m \leftarrow \min_{j=1..K} x_{ij};\quad x_i^d \leftarrow x_i^M - x_i^m$
  end for
  {Normalize the point}
  for $i = 1$ to $D$ do
    $x'_i \leftarrow (x_i - x_i^m)/x_i^d$
  end for
  return $x'$
```

Algorithm 3 ComputeWeights(X)

```
Input: $X$ – a cluster of normalized points.
Output: $w$ – a vector of weights of features.
  $s \leftarrow \sum_{i=1}^{D} \sigma_i$
  {where $\sigma_i$ is the standard deviation of feature $i$ in the cluster $X$}
  for $i = 1$ to $D$ do
    $w_i \leftarrow 1 - \sigma_i/s$
  end for
  return $w$
```

inequality (1) to normalized features, as described in Algorithm 2, we have

$$\frac{p_o - x_o^m}{x_o^d} > \frac{1}{K}\sum_{i=1}^{K}\frac{x_{oi} - x_o^m}{x_o^d} + 3\sigma'_o, \tag{2}$$

where

$$\sigma'_o = \sqrt{\frac{1}{K}\sum_{i=1}^{K}\left(\frac{x_{oi} - x_o^m}{x_o^d} - \frac{1}{K}\sum_{j=1}^{K}\frac{x_{oj} - x_o^m}{x_o^d}\right)^2}.$$

An analogous reasoning can be applied to the inequality $p_o - s_o < -3\sigma_o$.

3.4 Computation of Weights

Each indicator within a cluster is assigned a weight. That is, the same indicators of different clusters may have different weights. We have found that the weight of a feature should be dependent on the standard deviation of the feature within the corresponding cluster and on the standard deviations of other features. Therefore, if values of a feature are similar in different points of a cluster, the weight of that feature should be higher. This is shown in Algorithm 3. Having computed the weights of the indicators, the weight of an exchange/ATS is computed as the average of the weights of its indicators.

Algorithm 4 Predict(x)

```
Input: $x$, a normalized $D$-dimensional feature point to be classified.
       $d$, a distance threshold.    $r$, distance ratio threshold
Output: Either “predict price change up” or “predict price change down” or “not classified”.
  {Compute the squared distances to the centroids $c^d$ and $c^u$ of clusters down and up respectively,
  where $w_i^d$ and $w_i^u$ are the weights of the $i$-th indicator in the corresponding clusters.}
  $d_d \leftarrow \sum_{i=1}^{D} w_i^d (x_i - c_i^d)^2$;    $d_u \leftarrow \sum_{i=1}^{D} w_i^u (x_i - c_i^u)^2$

  if $d_d < d$ and $d_d/d_u < r$ then
    return “predict a price change down”
  end if
  if $d_u < d$ and $d_u/d_d < r$ then
    return “predict a price change up”
  end if
  return “not classified”
```

4 The Classifier

Quotes can be received from the market at a high frequency, often several quotes in a millisecond. To analyze incoming samples in real-time, the recognition model should be computationally efficient. For the classifier, we have chosen to compute the feature-weighted distance from a test point to the centroid of a cluster, since this is one of the least expensive techniques in artificial intelligence. The method compares patterns with a “golden mean”, the average point in the cluster that hypothetically represents the perfect combination of values of the indicators. There is not necessarily any actual data point with this value.

This technique is fast in training and classification. To *train* the model, one needs to collect a certain number of points in clusters and find the centroid of each cluster. Once the model is trained, it continues processing quotes and classify them. We are interested in two classes of points: one class for price changes up and another class for price changes down. *Classification* of a quote is performed by computing the squared weighted Euclidean distance from the feature-vector of the quote to the centroid of each training cluster, as shown in Algorithm 4.

As can be observed in the algorithm, we differentiate the notions of classification and prediction. Classification happens with each quote received – a feature vector is formed and the distances to centroids are evaluated. In contrast, a *prediction* is made only if the distances between the sample and the centroids satisfy certain criteria, i.e. if the feature point is relatively close to one of the two centroids. The prediction accuracy can be increased and the number of predictions decreased by reducing the thresholds d and r in Algorithm 4. The necessary values of the thresholds can be determined empirically or assigned heuristically.

Besides efficiency, the described model has other important features:

- *Adaptability*: If values of some of the indicators change, the centroid will slowly move in the direction of change. The coordinates of a centroid can be updated in constant time with each new point or a group of points.
- *Transparency*: The method described facilitates control of the impact that certain indicators or weights have on the distances. The values of features and their weights can be easily analyzed by human experts to validate the model.
- *Presence of a confidence measure*: The classification confidence can be derived intuitively from the distances and their ratios to regulate the accuracy and the number of predictions.

5 Complexity Analysis

The time complexity to compute the indicators may be expressed using an online complexity model [4] as $\mathrm{OL}_n[O(1), 0]$. That is, all of the indicators described in 3.1 may be computed in constant time as each event is received, and no additional time is required after the last event is received.

To check if a point is an outlier, given that the centroid and the standard deviation of the cluster have already been computed, it takes $O(D)$ time, D being the dimension of the feature space. To normalize a point, assuming that the maximum and the minimum have been found in each dimension, and to classify a point it takes $O(D)$ time as well. Computation of weights of features can also be done in $O(D)$ time.

Computation of indicators, removal of outliers, normalization, and computation of weights can be implemented in *parallel*: each dimension in a separate thread. This can yield further decrease in execution time. In practice, however, parallel computing resources might be better deployed using processors to watch different securities.

6 Experiments

6.1 Experimental Setting

The experiments were performed on data for MSFT (Microsoft) securities, using quotes from the following exchanges/ATSs: NYSE Archipelago Exchange (Arca), Better Alternative Trading System (BATS) BZX Exchange, BATS BYX Exchange, Chicago Board Options Exchange (CBOE), Direct Edge A (EDGA), Direct Edge X (EDGX), NASDAQ, NASDAQ OMX BX, National Stock Exchange, and NASDAQ PHLX. The recorded events include: change in bid/offer prices and bid/offer depth. We recorded several days in December, 2011 with the total of 9,389,993 quotes and 4,658 price changes. Training was performed until both clusters had at least 10 points. The value of the weight in computation of the *ROC* was taken as 0.6. After 5 changes in price, parameters of a cluster were recomputed.

We calculated two measures: *on-change* distance accuracy and *prediction* accuracy. The on-change distance measure was counted as correct if the distance to the centroid of the cluster in the direction of the price change was smaller than the distance to the other cluster. In other words, if a change down (up) was recorded and $d_d < d_u$ ($d_d > d_u$), the on-change distance measure was counted as correct.

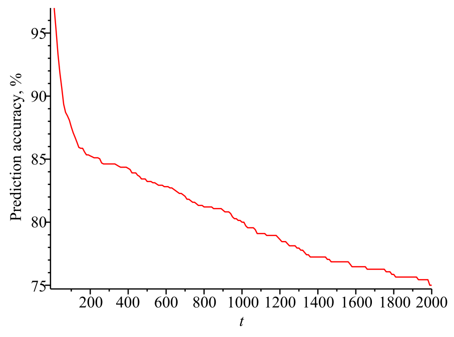
The prediction accuracy with a prediction interval of t was computed as follows. If the prediction was in the direction of the price change, and the interval between a prediction and the actual change was greater than t, the count of correct predictions was increased by one. If the interval was less than t, the count was not changed. If the change happened in the opposite direction, the count of wrong predictions was increased by one, independently of the time interval after the prediction. This measure aimed to simulate real-life trading, when execution of a transaction takes a certain amount of time, depending on infrastructure.

6.2 Experimental Results

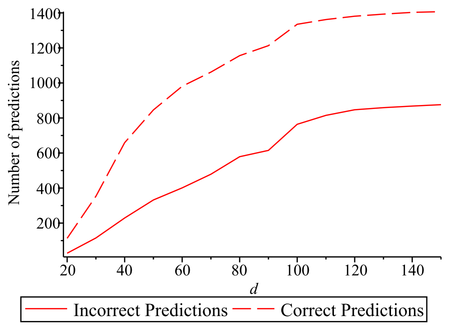
The on-change accuracy of the model on the recorded data was 96.25%.

The prediction accuracy as a function of t is presented in Figure 1(a) for the distance thresholds $d = 20$ and $r = 1/20$. There are 28 incorrect predictions for any t.

We also measured the number of correct and incorrect predictions depending on the distance threshold d with fixed $r = 1/20$ for $t = 1000ms$. The results are presented in Figure 1. Figure 2 shows the prediction accuracy as the function of d.



(a)



(b)

Figure 1: Prediction results:
(a) Prediction accuracy as a function of t (in milliseconds), and
(b) The number of correct and incorrect predictions depending on the distance threshold d

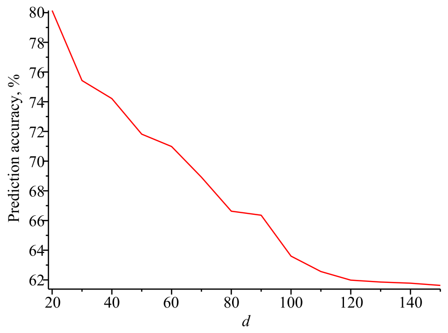


Figure 2: Prediction accuracy as a function of the distance threshold d

7 Conclusion and Future Work

A method for stock market analysis has been developed that is based on computation of the distance to the centroid of a set of feature vectors. The centroid represents an empirical combination of values of indicators that signal a price change. The model demonstrates good performance, even with naïve indicators.

For future work, one could consider additional market indicators to be added to the model (including those computed from correlated products) to increase the prediction interval. This however would trigger a slight increase in the processing time. A careful analysis is necessary to find the balance between the processing time, the prediction accuracy and the prediction interval. We expect that the model presented can be used to achieve balance.

Another direction for development of the model is a study of the dynamics of distances. We observe that price changes occur not only when the test point moves closer to the closer cluster, they sometimes occur only when the test point quickly moves farther from the farther cluster. To analyze these patterns, the sequence of distance values can be represented as a function. This function can then be approximated. The parameters of approximation will be used to describe the relationship between the sequences and therefore make well-grounded decisions.

Another direction is investigation of grouping of points within a cluster and dividing them in sub-clusters. Then a prediction will be based on proximity of a test point to one of the sub-clusters.

References

[1] Pei-Chann Chang, Chen-Hao Liu, Jun-Lin Lin, Chin-Yuan Fan, and Celeste S. P. Ng. A neural network with a case based dynamic window for stock trading prediction. *Expert Syst. Appl.*, 36:6889–6898, April 2009.

[2] Christopher Chatfield. *Time-series forecasting*. CRC Press, 2001.

[3] M. A. H. Dempster and C. M. Jones. A real-time adaptive trading system using genetic programming. *Quantitative Finance*, 1(4):397–413, 2001.

[4] Oleg Golubitsky and Stephen M. Watt. Online stroke modeling for handwriting recognition. In *Proceedings of the 2008 conference of the center for advanced studies on collaborative research: meeting of minds*, CASCON '08, pages 6:72–6:80, New York, NY, USA, 2008. ACM.

[5] G. Hughes. On the mean accuracy of statistical pattern recognizers. *Information Theory, IEEE Transactions on*, 14(1):55–63, January 1968.

[6] Xiaowei Lin, Zehong Yang, and Yixu Song. Short-term stock price prediction based on echo state networks. *Expert Systems with Applications*, 36(3):7313–7317, 2009.

[7] A Lora, J Santos, A Exposito, J Ramos, and J Santos. Electricity Market Price Forecasting Based on Weighted Nearest Neighbors Techniques. *Power Systems, IEEE Transactions on*, 22(3):1294–1301, 2007.

[8] Burton Malkiel. *A Random Walk Down Wall Street*. W. W. Norton & Company, Inc., 1973.

[9] Rogemar S. Mamon and Robert James Elliott. *Hidden Markov models in finance*. Springer, 2007.

[10] Ilya Raykhel and Dan Ventura. Real-time automatic price prediction for ebay online trading. In *in Proceedings of the Innovative Applications of Artificial Intelligence Conference*, pages 135–140, 2009.

[11] Wei Shen, Yunyun Zhang, and Xiaoyong Ma. *Stock Return Forecast with LS-SVM and Particle Swarm Optimization*, pages 143–147. IEEE, 2009.

[12] Stephen J. Taylor. *Modelling Financial Times Series*. World Scientific Publishing Company, 2 edition, December 2007.

[13] Yudong Zhang and Lenan Wu. Stock market prediction of s&p 500 via combination of improved bco approach and bp neural network. *Expert Systems with Applications*, 36(5):8849–8854, 2009.