Neurocomputing 72 (2009) 3439–3446

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

# A combination of hidden Markov model and fuzzy model for stock market forecasting

Md. Rafiul Hassan *

*Department of Computer Science and Software Engineering, The University of Melbourne, Australia*

ARTICLE INFO

Available online 21 June 2009

*Keywords:*
Hidden Markov model
Fuzzy logic
Stock market
Forecasting
Time series

ABSTRACT

This paper presents a novel combination of the hidden Markov model (HMM) and the fuzzy models for forecasting stock market data. In a previous study we used an HMM to identify similar data patterns from the historical data and then used a weighted average to generate a 'one-day-ahead' forecast. This paper uses a similar approach to identify data patterns by using the HMM and then uses fuzzy logic to obtain a forecast value. The HMM's log-likelihood for each of the input data vectors is used to partition the dataspace. Each of the divided dataspaces is then used to generate a fuzzy rule. The fuzzy model developed from this approach is tested on stock market data drawn from different sectors. Experimental results clearly show an improved forecasting accuracy compared to other forecasting models such as, ARIMA, artificial neural network (ANN) and another HMM-based forecasting model.

## 1. Introduction

The stock market is one of the most attractive investment choices, from which large profits can be earned. Forecasting stock market behavior has been always a challenging task due to its complex, volatile and non-linear nature [1]. Conventional statistical techniques for forecasting are constrained by the underlying seasonality, non-stationarity and other factors of the market [2,3]. Several Artificial Intelligence (AI) techniques, e.g. artificial neural networks (ANNs) [4,5], fuzzy logic (FL) [6,7] and support vector machines (SVMs) [8], have been proposed in the literature as a way of obtaining a better forecast. Of the existing AI methods, ANNs are effective in realizing the input–output mapping even when the exact relationship between the input and output is not known or is hard to detect mathematically [9]. This characteristic makes it particularly useful as a stock market forecaster. Nonetheless, the performance of ANN is constrained, in that it needs a lot of training data to be able to forecast more efficiently and accurately. Furthermore, 'black box' models like ANNs do not offer insight into the nature of the dataset when applied in financial time series forecasting and this poses difficulties in terms of fine-tuning the model. In contrast to ANN, the kernel-based hyper plane separation techniques used in SVMs have become popular as a classifier rather than as a forecasting tool.

Compared to the ANN, FL offers a clear insight into the model. FL is especially popular for dealing with non-linear systems and hence it is suitable for forecasting the stock market. Nevertheless, the performance of the method depends on the fuzzification of the time series data. The generation of appropriate fuzzy rules for a financial data is often challenging. For instance, a dataset with $n$ dimensions will produce a maximum of $n^n$ rules.

Hidden Markov models (HMMs) have been used in analyzing and predicting time series phenomena. They have been used extensively in areas like speech recognition [10,11], DNA sequencing [12], and ECG analysis [13]. Somewhat surprisingly, their use in predicting stock market time series has been limited. One study by Shi and Weigend [14] introduced an HMM to predict changes in the trajectories of financial market time series data. In related work, Zhang [15] modified the HMM training scheme to improve the prediction accuracy of Shi and Weigend's model.

In a recent study [3] the author developed a fusion model, by combining an HMM with an ANN and a GA, to generate one-day-ahead forecasts of stock prices. In that model, the optimized HMM was used to identify similar data patterns from the historical data. Then two alternative methods were examined for constructing the 'one-day-ahead' forecast value based on the HMM identified patterns. In the first approach, the neighboring values of the identified data patterns were interpolated to obtain the forecast value. In the second approach, a number of similar data patterns were identified from the historical data and then a weighted average of the differences in the values between the neighboring data items to the identified data was added to the current value to generate the corresponding forecast value.

In this paper, we present FL in combination with HMM in order to notably improve the prediction accuracy for non-stationary and non-linear stock market datasets. The FL is introduced here to

* Tel.: +61 03 8344 1349; fax: +61 03 9348 1184.
*E-mail address:* mrhassan@csse.unimelb.edu.au

0925-2312/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2008.09.029

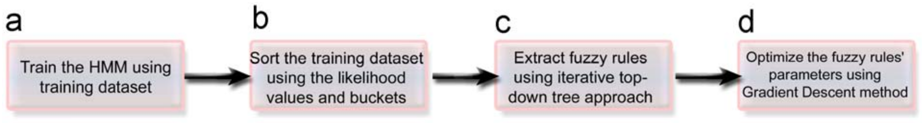
a
Train the HMM using training dataset
b
Sort the training dataset using the likelihood values and buckets
c
Extract fuzzy rules using iterative top-down tree approach
d
Optimize the fuzzy rules' parameters using Gradient Descent method

**Fig. 1.** The block diagram of the HMM–fuzzy model.

generate the one-day-ahead forecast value, while using the HMM to identify similar data patterns and group them based on these similarities. The approach combines the HMM's data pattern identification method to partition the dataspace with the generation of fuzzy logic for the prediction of multivariate financial time series data. Parameters of the generated fuzzy rules are further tuned by using a gradient descent algorithm. The forecast accuracy for stock data using the combination of the HMM and fuzzy model clearly outperforms the base fusion model [3].

The remainder of this paper is organized as follows. Section 2 introduces fuzzy rules and reviews some of the existing techniques for generating fuzzy rules. In Section 3, details of the combination of HMM and fuzzy model are described including the algorithms. Section 4 provides experimental results. Finally, in Section 5, we discuss the method and conclude the paper.

## 2. Fuzzy logic and fuzzy rule

Fuzzy logic typically processes non-linear datasets by mapping input data (feature) vectors into scalar output: i.e. it maps numbers into numbers. FL handles non-linearity well because of the fuzzy rules it uses to map the non-linear relationship between inputs and outputs. Fuzzy rule generation is usually based on past knowledge and experience. There are problems for which it is hard to find such knowledge or experience; however, data pertaining to solutions might be available. Data-driven fuzzy models generate fuzzy rules from available data, though success rates usually vary.

There exist several methods that generate fuzzy rules from datasets [16,17]. In order to efficiently generate fuzzy rules, a compaction of the complete system is required and this is usually achieved by minimizing the number of rules without negatively affecting overall performance. Compacting fuzzy models brings into play an influential factor: the partitioning of the input space [18]. Clustering is one method used to partition dataspaces to generate fuzzy rules. Grid partitioning and tree partitioning are also used in data-driven fuzzy models to generate fuzzy rules from the available datasets. While generating fuzzy rules based on clustering, the performance of the model depends on the appropriate number of clusters chosen prior to the generation of the model. In contrast, the grid partitioning method suffers from exponential rule explosion with an increasing of the number of input features. The tree partitioning method produces more than one membership function (MF) for each of the input features, which is responsible for generating a large number of rules. MF is a mathematical function that is used to impose the membership degree/level/involvement of an input feature to a fuzzy set.

All the existing rule generation methodologies that rely on dataspace partitioning work on the presumption that the total number of rules to be generated is determined before the actual building process commences. This paper presents the development of an automated fuzzy rule generation method that does not require the number of clusters/partitions in the dataset to actually start building a data-driven fuzzy model. In the subsequent section we describe the proposed fuzzy model generation method.

## 3. The HMM–fuzzy combination

This section describes the development of a fuzzy generation method based on the HMM-log-likelihood value from each of the data patterns in the dataset. This method is unique, in that the number of clusters/partitions is not required prior to the fuzzy rule generation. On the other hand, it does require us to provide our desired error margin in the performance of the final fuzzy model.

From a high level of abstraction, it may be said that the HMM is used to partition the input space depending on the similarities in the input data sequences. An HMM is trained using the training dataset and then the HMM-log-likelihood values calculated in the training stage (Fig. 1(a)) are ordered using a number of 'buckets' or 'bins'. Each of the 'buckets' contains similar patterns (Fig. 1(b)) that produce log-likelihood values within the range of the bucket's start point and end point. A recursive divide and conquer algorithm (Fig. 1(c)) is then used to generate a set of fuzzy rules. Finally, further optimization of the fuzzy rule parameters is done using a gradient descent method (Fig. 1(d)).

### 3.1. Likelihood values using the HMM

An initial HMM structure is built and the parameter values are re-estimated for a given dataset. It may be noted here that datasets fed into the HMM are usually arranged in an order so that each data vector forms a pattern. For a multivariate data, each data vector must contain all the predictor/independent variables. To explain this using a hypothetical situation, let us consider the time series for the daily sales of a product say, a fizzy beverage. The sales figure, *amount*, of units sold may depend on factors as diverse as the weather, *weather*, the quality of the product, *quality*, competition marketing and sales, $sale_{other}$, etc. In this case, the set of predictor variables is $weather, quality, sale_{other}$. The values of these variables for each time unit would create a data vector for that particular time. At this stage the sequence of variables in each of the data vectors is assumed to appear consecutively, almost as if each data vector forms specific patterns, as shown in Fig. 2. Once the dataset is arranged in this manner, they are fed into the HMM to re-estimate the parameter values. The initial HMM is built and it is trained using the training dataset as per the procedure in the study by Hassan et al. [3]. It should be noted that, for the stock data considered in this paper, the following data features are used to form the data vector: daily *opening*, *high*, *low* and *closing* price.

Once the HMM is trained, this HMM is used to generate a log-likelihood value for each of the data vectors in the dataset. To generate the log-likelihood values the forward algorithm [19] is used. For a total $m$ number of data vectors/patterns $m$ log-likelihood values are generated, each labeled with the index of the corresponding data vector in the dataset.

**Example 3.1.** Let us consider a dataset $\mathfrak{D}$, where $\vec{x}_i \in \mathfrak{D}$ for $i = 1, \ldots, m$, where, $i$ represents the index for data vector $\vec{x}_i$ and $x_{i_j} \in \vec{x}_i$ for $j = 1, \ldots, k$. That is, each of the data vectors is $k$-dimensional and the dataset contains $m$ data vectors. Assume that the dataset $\mathfrak{D}$ represents a stock data: i.e. the $i$th data vector $\langle x_{i_1}, x_{i_2}, x_{i_3}, x_{i_4} \rangle$ will be $\langle x_{i_{open}}, x_{i_{high}}, x_{i_{low}}, x_{i_{close}} \rangle$. For this dataset we have the trained HMM $\lambda$. According to Rabiner [20] the value of $\log(\Pr(\vec{x}_i|\lambda))$, called log-likelihood, represents the possibility to

|  | Weather | Quality | Sale (Other) |
|---|---|---|---|
| Day 1 | 26 | 4 | 20 |
| Day 2 | 24 | 3 | 44 |
| Day 3 | 36 | 2 | 90 |
| Day 4 | 23 | 4 | 16 |

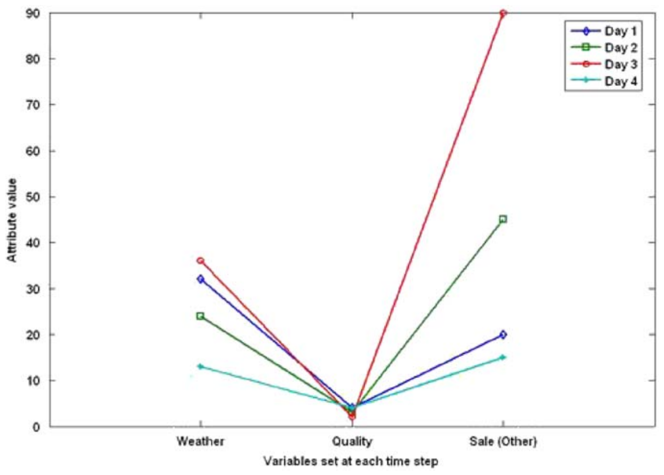

**Fig. 2.** The data pattern formed by each of the data vectors.

| Data vector index i | $j = 1$ *Open* | $j = 2$ *High* | $j = 3$ *Low* | $j = 4$ *Close* |
|---|---|---|---|---|
| 1 | 10.5 | 11.25 | 10 | 10.75 |
| 2 | 8 | 9.5 | 8 | 8.5 |
| 3 | 10.5 | 11.25 | 9.5 | 11 |
| ......................................... | | | | |
| ......................................... | | | | |
| $m$–2 | 8.5 | 9.25 | 8.25 | 8.25 |
| $m$–1 | 10.75 | 11.5 | 10.5 | 11 |
| $m$ | 7.5 | 9 | 7.5 | 7.75 |

**Fig. 3.** Dataset labeled with its record number as index.

| Data vector index i | loglikelihood |
|---|---|
| 1 | –0.0010 |
| 2 | –3.0080 |
| 3 | –0.0008 |
| .......................... | |
| .......................... | |
| $m$–2 | –3.0090 |
| $m$–1 | –0.0009 |
| $m$ | –3.0060 |

**Fig. 4.** Each data vector is represented by its index and the corresponding log-likelihood value.

generate the data vector/pattern $\vec{x}_i$ by the $\lambda$. In our method we generate a log-likelihood value $l_i$ for each of the data vectors $\vec{x}_i$ in $\mathfrak{D}$. The set of generated log-likelihood values for the dataset in Fig. 3 is shown along with the data vector indices in Fig. 4. Thus, Fig. 4 represents the labeling of log-likelihood values with the index of the corresponding data vector in the dataset.

### 3.2. Grouping of similar data sequences

The range of log-likelihood values ($l_1$–$l_m$, $l_i =$ log – likelihood value produced for the ith data vector) is split

```
1. bucket_size=$\theta$;
2. startRange= minimum of the likelihood values;
3. endRange= maximum of the likelihood values;
4. i=startRange;
5. j=1;
6. while (i $\le$ endRange)
   a     bucket[j].start=i;
   b     bucket[j].end=i+bucket_size;
   c     bucket[j].data = find (dataLikelihood $\ge$i and dataLikelihood < i+bucket_size)
   d     i=i+bucket_size;
   e while end
```

**Fig. 5.** The pseudo-code to split the range of log-likelihood into buckets/bins.

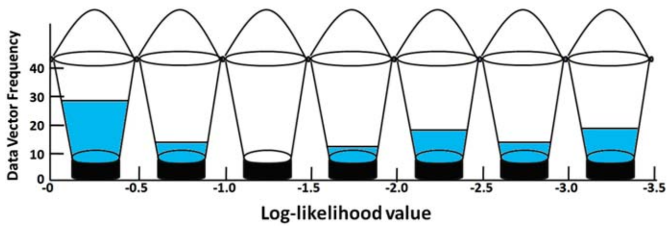

**Fig. 6.** The bucketing approach to group data with similar log-likelihood values.

into equal sized buckets/bins (Figs. 5 and 6). The data vectors in each bucket produce similar log-likelihood values. Figs. 7 and 8 show an example of dividing a number of data patterns into two buckets. We note that the data patterns belonging to the bucket of Fig. 7 produce similar log-likelihood values within the vicinity of a range of log-likelihood values, while they differ from those belonging to the bucket of Fig. 8.

Each of the buckets has a starting point and an ending point corresponding to the log-likelihood values. The size of the bucket, $\theta$, is a parameter of the model that is used to guide the rule extraction process at a later stage. For instance, in Figs. 7 and 8 the size of each bucket is 0.5 (i.e., $\theta = 0.5$). The starting point for the bucket in Fig. 7 is $-3.5159$ and the ending point is $-4.0159$, whereas the starting point for the bucket in Fig. 8 is $-4.5159$ and the ending point is $-5.0159$. Details of the bucketing algorithm are depicted in Fig. 5.

**Example 3.2.** Let us consider the dataset described in Example 3.1. For this dataset the range of log-likelihood value is between $\min_{\forall i}(abs(l_i))$ to $\max_{\forall i}(abs(l_i))$, where $l_i =$ log-likelihood value for the data vector with index $i$. As shown in Fig. 4, the range would be $-0.0008$ to $-3.0090$. We choose $\theta = 0.5$. Thus there will be a total of seven buckets as shown in Fig. 6. The figure also reveals that there are 28 data vectors that generated log-likelihood values in the range of $-0.0$ to $-0.5$. That is in terms of the log-likelihood value these 28 vectors exhibit a similar pattern.

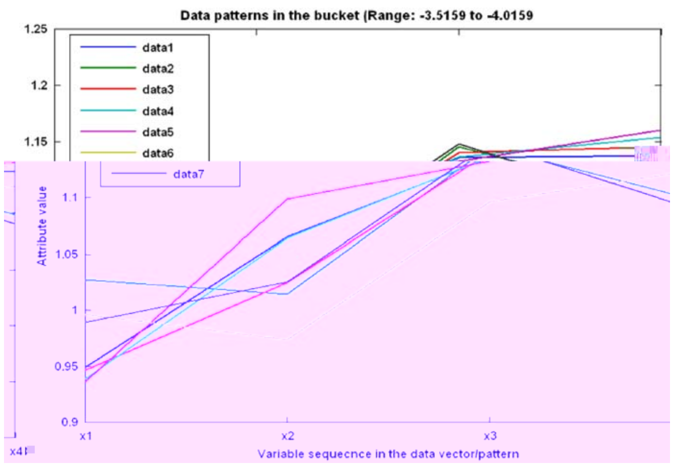

**Fig. 7.** The bucketed data vectors in a specific bucket.

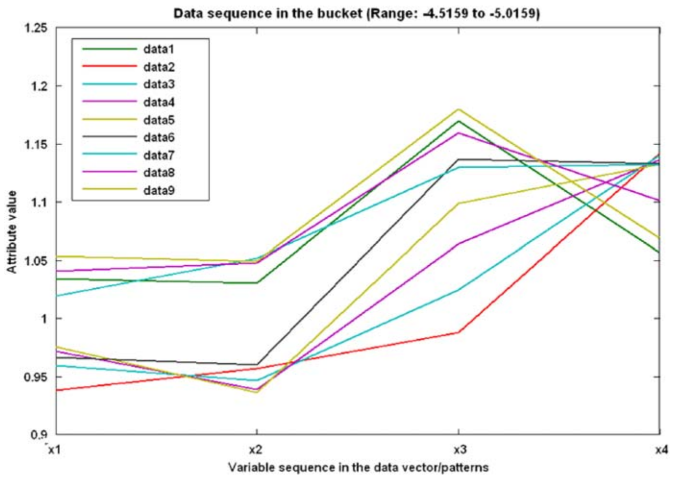

**Fig. 8.** The bucketed data vectors in a different bucket than the bucket in Fig. 7.

## 3.3. *The fuzzy model*

Once buckets have been created the fuzzy rule extraction begins. Initially, rules are generated from the dataset. The rule generation process is briefly described below, followed by insight into the proposed top-down tree algorithm for the rule extraction.

For the consequent part of the fuzzy rules, the TS [21] model is used in which the output is calculated as the linear combination of weighted input variables. As described earlier in this paper, the inference in the TS model is made from the fuzzy rule as follows: if $x$ is $M(x)$ and $y$ is $M(y)$ then $z = p_1x + q_1y + r_1$, where $M(x)$ and $M(y)$ are membership functions and $p_1$, $q_1$ and $r_1$ are linear parameters. The weight $W$ of this rule in the fuzzy model is: $W = \max(M(x), M(y))$.

### 3.3.1. *Fuzzy rule extraction*

For the antecedent part of the fuzzy rules, our approach chose the Gaussian membership function. So, for $k$ variables in a data vector/pattern, there exist $k$ membership functions. For each of the input variables, the mean $\mu_{fuzzy}$ and the standard deviation $\sigma_{fuzzy}$ are calculated. Then, the $k$th (for the $k$th input variable) membership function is

$$M_i(x_k) = e^{(-1/2)((x_k-\mu_{fuzzy_{ik}})/\sigma_{fuzzy_{ik}})^2} \tag{1}$$

where $M_i(x_k)$ is the membership function for rule $i$ and data feature $k$, $\mu_{fuzzy_{ik}}$ the mean or the location of the $k$th membership function in rule $i$, and $\sigma_{fuzzy_{ik}}$ the standard deviation or the steepness of the $k$th membership function in rule $i$.

Following the standard representation of the fuzzy rule, the generated $i$th rule in linguistic form would be as: if $x_1$ is $M_i(x_1)$ and ... and $x_k$ is $M_i(x_k)$ then (output) $p_{i1}x_1 + p_{i2}x_2 + \cdots + r_i$ $(p_{i1}, p_{i2}, \ldots$ ., $r_i$ are linear parameters). A graphical representation of such two fuzzy rules along with the de-fuzzification process is shown in Fig. 9.

### 3.3.2. *The top-down tree approach to generate the fuzzy rules*

At the beginning, just one fuzzy rule is created to represent the entire input space of the training dataset. At this point, all log-likelihood values contained in the individual buckets may be perceived as belonging to one global bucket. This rule is then used to predict the variable of interest (the next day's closing price for the stock data). The level of the rule's efficacy is determined based on the prediction error for the training dataset, which is in turn calculated in terms of mean squared error (MSE).

If the prediction error for the training dataset is less than or equal to a threshold value $\xi$, the algorithm is terminated and no further rules are extracted. On the other hand, if the prediction error is greater than $\xi$, then the input space is split into two parts

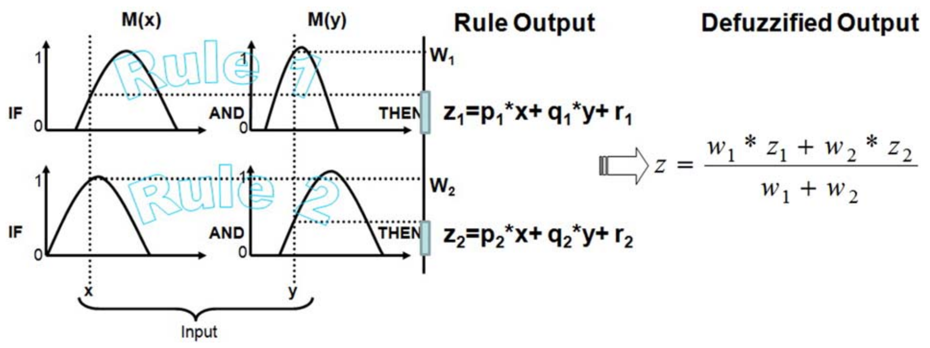

**Fig. 9.** A graphical representation of the TS fuzzy rule ($W_i$ represents the weights of each of the rules; typically the weight of a rule is chosen as the maximum value among the membership values of all input variables, i.e. this weight for rule 1 is $W_1 = \max(M(x), M(y))$.

```
Error_Threshold = $\xi$; ($\xi$ = desired error value)
Extract only one rule using the whole training data set
error = calculate_error (data, rules);
if (error > error threshold $\xi$)
    divide the total number of bucket sinto two equal part and extract rules for each
of the part
    error = calculate_error (data, rules);
    left_flag = true
    right_flag = false
end if
while (error > error threshold $\xi$)
    if (left_flag = true)
      divide the left part into two equal part using bucket number and
      extract rules for each of the part
      error = calculate_error (data, rules);
      left_flag = false
      right_flag = true
    else
      divide the right part into two equal part using bucket number
      extract rules for each of the part
      error = calculate_error (data, rules);
      left_flag = true
      right_flag = false
    end if
end while
return rules
_____________________________________________
function error = calculate_error (data, rule)
simulate result using extracted rule
error = mse (produced_output-actual_output)
return error;
```

**Fig. 10.** The algorithm for rule extraction using buckets.

with the help of the buckets produced in Section 3.2 (see Fig. 5). The splitting of the input space is achieved by dividing the total buckets into two equal parts. Data in the respective parts constitute the divided input space. Each divided partition has individual rules created for it. Finally, the total number of rules is increased by one. The prediction error for the training dataset is recalculated using the extracted rule set. Should the error threshold $\xi$ not be reached then the buckets containing the datasets responsible for the left part of the rule are divided into two rules, and the process is iterated. Again, if the error threshold $\xi$ is still not met, the right part of the rule is partitioned and the process is undertaken again. This cycle continues until either the error threshold $\xi$ is met or the number of rules equals the number of buckets. The pseudocode in Fig. 10 describes the process for the rule extraction.

**Example 3.3.** Let us consider the dataset described in Example 3.1 and the generated seven buckets of data vectors in Example 3.2. First, a fuzzy rule is generated using all the data vectors in dataset $\mathfrak{D}$. This fuzzy rule would be as follows: *if $x_{open}$ satisfies $M_{open}$ with linear parameter $p_1$ and $x_{high}$ satisfies $M_{high}$ with linear parameter $p_2$ and $x_{low}$ satisfies $M_{low}$ with linear parameter $p_3$ and $x_{close}$ satisfies $M_{close}$ with linear parameter $p_4$ then predict$_{close}$ is $p_1 \times x_{open} + p_2 \times x_{high} + p_3 \times x_{low} + p_4 \times x_{close}$.*

We assume that the prediction error for the generated fuzzy rule is 0.9, while the desired minimum error $\xi$ is chosen as 0.4. To meet $\xi$ the number of fuzzy rules is increased by one. To do so, the dataset is divided into two parts using the seven buckets. The first divided part consists of the data vectors which generated a log-likelihood value in the range of $-0.0$ to 1.5 (i.e. buckets 1–3) and the second divided part contains the data vectors which generated log-likelihood values in the range of $-1.5$ to 3.5 (i.e. buckets 4–7). Now, two fuzzy rules are generated using these two parts of the divided dataset. Now we obtain a prediction error 0.3 using the two fuzzy rules. As the prediction error is less than $\xi$, further rule generation is stopped at this step.

### 3.4. Optimization of extracted fuzzy rules

To optimize parameters for the extracted fuzzy rules, a gradient descent algorithm is employed. The objective here is to minimize the MSE for the training dataset. In the TS fuzzy model, every dataset has two parameters that need to be optimized. The first one is the non-linear (premise) parameter, and the second one is the set of linear (consequence) parameters. In the proposed model the optimization technique ANFIS, presented by Jang [22], is used, where a gradient descent method along with the least squared error (LSE) estimate is applied.

## 4. Experiment and results

To investigate the performance of the combination of the HMM and the fuzzy model, forecasts are generated for six stocks drawn from the airlines sector and IT sector, respectively: namely British Airlines, Delta Airlines, Ryanair Airlines, Apple Computer Inc., International Business Machines Corporation (IBM) and Dell Inc. For each company four attributes, opening, high, low and closing price from the daily stock market prices, are used to form the observation vector. The forecast variable is the next day's closing price. Descriptions and details of the experimental setup, datasets used, and the results obtained follow.

### 4.1. Stock market forecasting

#### 4.1.1. Dataset

The six stocks, mentioned above, were used in our previous study [3,1]. For the sake of consistency, in this experiment we have used the same training and test dataset for those six stocks. Details about the training and test dataset are given in Table 1.

#### 4.1.2. Experimental setup

For each of the stocks, the bucket size is set to be $\theta = 0.5$; the actual bucketing is performed with log-likelihood values and the allowable margin of error is set in the range of 0.02 to 0.002.

*Input data vector*: In our experiment the input data vector contains daily opening, high, low and closing prices to predict the next day's closing price. The reason for choosing these original prices compared to the technical indices is as follows:

There are many technical indices that can be used for stock market prediction, e.g. moving average, exponential moving average, weighted moving average, relative strength index, and high–low ratio. In fact, the technical index generalizes the relationship amongst the stock data (or the time series data) to predict the future trend of the time series. However, to predict the exact value of a time series the transformation of original (raw) data into technical format might not help, due to the loss of some valuable information depicted by the original data. That is, in the

**Table 1**
Training and test dataset.

| Stock type | Stock name | Training data | | Test data | |
|---|---|---|---|---|---|
| | | From | To | From | To |
| Airline sector | British Airlines | 17 September 2002 | 10 September 2004 | 11 September 2004 | 20 January 2005 |
| | Delta Airlines | 27 December 2002 | 31 August 2004 | 01 September 2004 | 17 November 2004 |
| | Ryanair Airlines | 06 May 2003 | 06 December 2004 | 07 December 2004 | 17 March 2005 |
| IT sector | Apple Computer Inc. | 10 February 2003 | 10 September 2004 | 13 September 2004 | 21 January 2005 |
| | IBM Corporation | 10 February 2003 | 10 September 2004 | 13 September 2004 | 21 January 2005 |
| | Dell Inc. | 10 February 2003 | 10 September 2004 | 13 September 2004 | 21 January 2005 |

**Table 2**
Parameter values used in HMM–fuzzy and ANN.

| Stock name | HMM–fuzzy | | | Artificial neural network | |
|---|---|---|---|---|---|
| | Desired MSE $\xi$ | Membership function | Bin size $\theta$ | No. of neurons in hidden layer | Desired MSE |
| British Airlines | 0.0016 | Gaussian | 0.5 | 8 | 0.001 |
| Delta Airlines | 0.2000 | Gaussian | 0.5 | 8 | 0.001 |
| Ryanair Airlines | 0.8000 | Gaussian | 0.5 | 4 | 0.001 |
| Apple Computer Inc. | 0.3000 | Gaussian | 0.5 | 2 | 0.001 |
| IBM Corporation | 0.9900 | Gaussian | 0.5 | 30 | 0.001 |
| Dell Inc. | 0.2700 | Gaussian | 0.5 | 30 | 0.001 |

**Table 3**
Forecast performance evaluation with that of previous HMM-based forecast model.

| Stock name | HMM-based forecasting model (MAPE) | Fusion HMM–ANN–GA with weighted average (MAPE) | Combination of HMM–fuzzy model (MAPE) |
|---|---|---|---|
| British Airlines | 2.629 | 1.646 | 1.529 |
| Delta Airlines | 6.850 | 5.529 | 4.535 |
| Ryanair Airlines | 1.928 | 1.377 | 1.356 |
| Apple Computer Inc. | 2.837 | 1.925 | 1.796 |
| IBM Corporation | 1.219 | 0.849 | 0.779 |
| Dell Inc. | 1.012 | 0.699 | 0.405 |

The mean absolute percentage error (MAPE) in the forecast for unseen test dataset.

**Table 4**
Forecast performance evaluation with that of ARIMA and ANN.

| Stock name | Combination of HMM–fuzzy model (MAPE) | ARIMA (MAPE) | ANN (MAPE) |
|---|---|---|---|
| British Airlines | 1.529 | 1.573 | 2.283 |
| Delta Airlines | 4.535 | 5.294 | 9.147 |
| Ryanair Airlines | 1.356 | 1.504 | 1.504 |
| Apple Computer Inc. | 1.796 | 1.801 | 1.801 |
| IBM Corporation | 0.779 | 0.972 | 0.972 |
| Dell Inc. | 0.405 | 0.660 | 0.660 |

The mean absolute percentage error (MAPE) in the forecast for unseen test dataset.

original data the interrelationship amongst the data can be better observed compared to the technical indices.

*Choice of parameters*: Following the study [1,3], the number of states in HMM for the stocks is chosen as four, as the number of input features is four (i.e. opening, high, low and closing) in the dataset. The initial parameter values of the HMM are chosen following the same steps as was chosen in study Hassan et al. [3]. The other parameters of the HMM–fuzzy model chosen in this experiment are listed in Table 2. To improve performance, we scaled the data between −1 and +1 before using them to predict the next day's closing price. Eq. (2) was used to scale each of the variables/features (e.g. $x_{open}$ is a feature) in the dataset:

$$x_{i_{scaled}} = 2 \times \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)} - 1 \qquad (2)$$

The parameter values for ARIMA were chosen by analyzing the ACF and PACF plot of the respective time series data. While using ANN in this experiment, the activation function for each of the neurons was chosen as a tan-sigmoidal function. A back propagation training algorithm was used to train the ANN. The maximum number of epochs for each of the stocks was set to 7000. A three-layer (one input layer, one hidden layer, and one output layer) feed-forward ANN was used to predict the next day's closing price. To obtain a better performance the number of neurons in the hidden layer was chosen by employing an evolutionary algorithm. Table 2 lists the number of neurons used for each of the stocks considered in this experiment.

### 4.1.3. *Performance metrics*

The performance metric used in this study is the mean absolute percentage error (MAPE) in accuracy. This value is calculated by first taking the absolute deviation between the actual value and the forecast value. Then the total of the ratio of deviation value with its actual value is calculated. The percentage of the average of this total ratio is the mean absolute percentage error. The following equation shows the process of calculating the MAPE.

Mean Absolute Percentage Error (MAPE)

$$= \frac{\sum_{i=1}^{r}\left(\dfrac{abs(y_i - p_i)}{y_i}\right)}{r} * 100\% \qquad (3)$$

where $r$ is the total number of test data sequences, $y_i$ the actual stock price on day $i$, and $p_i$ the forecast stock price on day $i$.

### 4.1.4. *Results*

Table 3 reports the forecast performance of the developed HMM–fuzzy model for the six stocks in terms of the forecast error using the MAPE metric. To investigate the performance improvement, the forecast performance obtained using the fusion HMM-based forecasting model developed in the previous study [1,3] is also presented. Table 4 compares the forecast accuracy of the developed model with that of ARIMA and ANN.

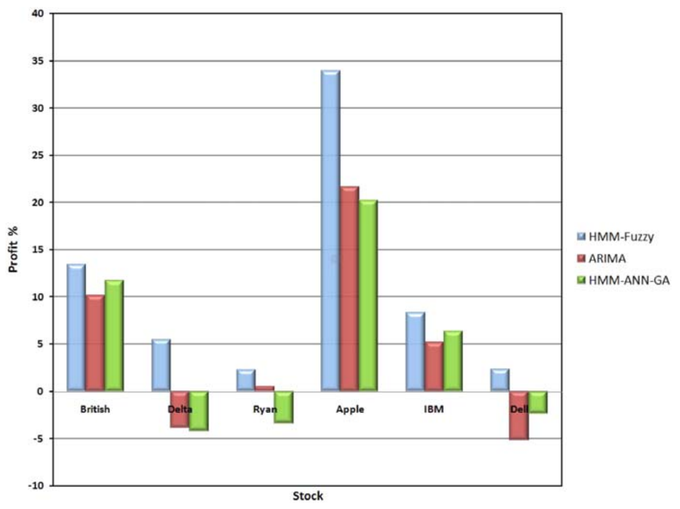

**Fig. 11.** Comparison of profit for the three models: HMM–fuzzy, ARIMA and HMM–ANN–GA for the six stocks.

A comparison of profit for the three models: HMM–fuzzy, ARIMA and HMM–ANN–GA for the six stocks is shown in Fig. 11. Here, the transaction decision is made based on forecasting of the stock price.

## 5. Conclusion

This paper introduces a novel combination of the HMM and the fuzzy model to forecast stock market data. The model makes use of the HMM's data partitioning approach to generate appropriate fuzzy rules. The HMM-based data partitioning technique includes the relationship between data features. This is because it uses the Markov process, where it is assumed that the current event/feature depends on the immediate past event/feature in the data vectors. Hence, the generated fuzzy model becomes suitable for a better performance. The top-down tree approach facilitates the identification of a 'reasonable' number of fuzzy rules for the non-linear data, in order that it meets the predefined MSE for the training dataset. Parameters of the generated fuzzy rules are further tuned using a gradient descent algorithm.

The developed model has been tested by preparing forecasts for the financial time series data of six stock prices. It is noticeable that the model produces improved forecasts for the stock datasets (see Table 3 for comparison with other HMM-based forecasting models). Among the other HMM-based forecasting models, the fusion model could produce a reasonable forecast performance, although the computational complexity of the model is huge as it uses a GA. For all six stocks considered in this paper, the developed model consistently performs better in terms of the MAPE. The profit also shows that HMM–fuzzy is more reliable than the other models.

It is interesting to note that the forecast accuracy of the combination HMM–fuzzy model is always better when compared to the ARIMA and ANN (see Table 4). While using ARIMA one has to be careful to choose the values of the parameters $p, q$ and $r$ ($p$ is the time lag in the time series, $q$ the differencing parameter, and $r$ the moving window size in the time series). In this paper, we have analyzed the ACF and PACF plots for the respective stock data to build the ARIMA. However, it is quite a task and almost impossible for a person without statistical knowledge to generate ARIMA for a problem. In contrast, it is easy and simple to use the proposed model and generate an appropriate fuzzy model for a problem. Furthermore, an improved forecast accuracy and higher profit are also the added benefit of using the proposed model when compared to the ARIMA.

In the HMM–fuzzy model, the initial weight for each of the generated fuzzy rules was chosen as 1. Every time the model gets a new data instance the membership values for each of the features for each rule is calculated and the maximum value among the membership values for a rule is considered as the weight for that rule. However, a further improvement in performance can be achieved if a better weighting scheme is developed where the weight of each rule will signify the importance of the rule in predicting output. We plan to develop a new adaptive weighting scheme to further fine-tune the generated fuzzy rules.

## Acknowledgments

The author would like to thank Dr. Noel Patson, lecturer, School of Computing, Central Queensland University, Australia and the anonymous reviewers for helpful and insightful comments and suggestions.

## References

[1] M.R. Hassan, B. Nath, Stock market forecasting using hidden Markov model: a new approach, in: Proceedings of the Fifth International Conference on Intelligent Systems Design and Applications, 2005, pp. 192–196.
[2] M.K. Tambi, Forecasting exchange rate a uni-variate out of sample approach (box Jenkins methodology), International Finance-Economics Working Paper Archive, 2005, p. 0506005.
[3] M.R. Hassan, B. Nath, M. Kirley, A fusion model of HMM, ANN and GA for stock market forecasting, Expert Systems with Applications 33 (1) (2007) 171–180.
[4] J.H. Choi, M.K. Lee, M.W. Rhee, Trading s&p500 stock index futures using a neural network, in: Proceedings of the Third Annual International Conference on Artificial Intelligence Applications on Wall Street, 1995, pp. 63–72.
[5] S. Wu, R. Lu, Combining artificial neural networks and statistics for stock-market forecasting, in: ACM Conference on Computer Science, 1993, pp. 257–264.
[6] E. Vercher, J.D. Bermudez, J.V. Segura, Fuzzy portfolio optimization under downside risk measures, Fuzzy Sets and Systems 158 (2007) 769–782.
[7] R. Simutis, Fuzzy logic based stock trading system, in: Proceedings of the IEEE/IAFE/INFORMS 2000 Conference on Computational Intelligence for Financial Engineering, 2000, pp. 19–21.
[8] L. Cao, F.E.H. Tay, Financial forecasting using support vector machines, Neural Computation and Application 10 (2001) 184–192.
[9] R.K. Begg, M.R. Hassan, Artificial neural networks in smart home, in: J.C. Augusto, C.D. Nugent (Eds.), Designing Smart Home, Springer, Berlin, 2006 (Chapter 9).
[10] H. Xie, P. Anreae, M. Zhang, P. Warren, Learning models for english speech recognition, in: Proceedings of the 27th Conference on Australasian Computer Science, 2004, pp. 323–329.
[11] F. Jelinek, M. Kaufmann, C.S. Mateo, Self-organized language modelling for speech recognition, in: A. Waibel, K.-F. Lee (Eds.), Readings in Speech Recognition, Morgan Kaufmann, San Mateo, CA, 1990.
[12] L.W.K. Cheung, Use of runs statistics for pattern recognition in genomic DNA sequences, Journal of Computational Biology (2004) 107–124.
[13] D.A. Coast, R.M. Stern, G.G. Cano, S.A. Briller, An approach to cardiac arrhythmia analysis using hidden Markov models, IEEE Transactions on Biomedical Engineering 37 (9) (1990) 826–836.
[14] A.S. Weigend, S. Shi, Predicting daily probability distributions of s&p500 returns, Journal of Forecasting (2000) 375–392.
[15] Y. Zhang, Prediction of financial time series with Hidden Markov Models, Masters Thesis, School of Computing Science, Simon Fraser University, 2004.
[16] J.S.R. Jang, C.T. Sun, E. Mizutani, Neuro-Fuzzy and Soft-Computing: A Computational Approach to Learning and Machine Intelligence, Prentice-Hall, Upper Saddle River, NJ, 1997.
[17] S.-M. Chen, S.-H. Lee, A new method for generating fuzzy rules from numerical data for handling classification problems, Applied Artificial Intelligence (2001) 645–664.
[18] J. Zurada, Optimal data driven rule extraction using adaptive fuzzy-neural models, Ph.D. Dissertation, University of Louisbille, 2002.
[19] A.B. Poritz, Hidden Markov models: a guided tour, in: Proceedings of ICASSP, 1988, pp. 7–13.
[20] L.R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, Proceedings of the IEEE 77 (1989) 257–286.
[21] T. Takagi, M. Sugeno, Fuzzy identification of systems and its application to modeling and control, IEEE Transactions on System, Man and Cybernetics (1985) 116–132.
[22] J.R. Jang, ANFIS: adaptive-network-based fuzzy inference system, IEEE Transactions on Systems, Man and Cybernetics 23 (1993) 51–63.

**Dr. Md. Rafiul Hassan** is working as a research fellow at the Department of Computer Science and Software Engineering in the University of Melbourne, Australia. He received a Ph.D. in 2007 from the University of Melbourne and a B.Sc. (Engg.) in Electronics and Computer Science in 2000 from Shah Jalal University of Science and Technology, Bangladesh. His research interests include neural networks, fuzzy logic, evolutionary algorithms, hidden Markov model, and support vector machine, with a particular focus on developing hybrid soft computing models for forecasting time series data. Before joining the University of Melbourne, Dr. Hassan worked as an assistant professor at the Department of Computer Science and Engineering in Shah Jalal University of Science and Technology, Bangladesh. He is currently involved in research and development projects for effective classification of Bioinformatics data. He is the author of around 20 papers in recognized international journals and conferences. He is a member of Australian Society of Operations Research (ASOR), IEEE and IEEE Computer society; and is involved in several Program Committees of international conferences. He also serves as the reviewer of a few renowned journals such as Information Science, Digital Signal Processing, and Computer Communications.