Evolving Hypernetwork Models of Binary Time Series for Forecasting Price Movements on Stock Markets

Elena Băutu, Sun Kim, Andrei Băutu, Henri Luchian and Byoung-Tak Zhang

Abstract— **The paper proposes a hypernetwork-based method for stock market prediction through a binary time series problem. Hypernetworks are a random hypergraph structure of higher-order probabilistic relations of data. The problem we tackle concerns the prediction of price movements (up/down) on stock markets. Compared to previous approaches, the proposed method discovers a large population of variable subpatterns, i.e. local and global patterns, using a novel evolutionary hypernetwork. An output is obtained from combining these patterns. In the paper, we describe two methods for assessing the prediction quality of the hypernetwork approach. Applied to the Dow Jones Industrial Average Index and the Korea Composite Stock Price Index data, the experimental results show that the proposed method effectively learns and predicts the time series information. In particular, the hypernetwork approach outperforms other machine learning methods such as support vector machines, naive Bayes, multilayer perceptrons, and k-nearest neighbors.**

I. Introduction

Forecasting a time series is a frequently encountered problem in many economic activities. Many factors influence the evolution of stock prices, such as political or social events, traders actions and economic conditions in general, therefore making predictions is an extremely challenging task. As far as making profit is concerned, apart from predicting the actual value of stock on the market, an important problem is to predict the direction of the market. Profitable market trading strategies may be constructed upon good predictions on the market direction. The direction of stock returns was proved to be predictable under some conditions [1]. Price movements on stock markets may be recorded as a binary time series, encoding the up/down movements as binary values. Binary time series are considered in many other practical situations when the occurrence of an event is recorded and needs to be predicted, such as the occurrence of meteorological phenomena (e.g. rainfalls [2]).

Here, we tackle the problem of forecasting a binary time series that models the increases and decreases in the price of certain stocks on the market. To solve this problem, we propose a novel approach based on hypernetwork models [3], which encompasses evolutionary techniques to learn the hypernetwork from data. The evolutionary training process of the hypernetwork described in the paper allows the evaluation of a large number of hyperedges by replacing those that contain obsolete information.

Elena Băutu, Andrei Băutu and Henri Luchian are with the Faculty of Computer Science, Al. I Cuza University, Iaşi 700483, România (email: ebautu@univ-ovidius.ro; abautu@anmb.ro; hluchian@infoiasi.ro). Sun Kim and Byoung-Tak Zhang are with the School of Computer Science and Engineering, Seoul National University, Seoul 151-744, Korea (email: skim@bi.snu.ac.kr; btzhang@bi.snu.ac.kr).

The proposed method identifies a large set of time series patterns of variable size in data and stores them as hyperedges. In hypernetworks, the hyperedges with large order represent specialized information which can be used locally, only in a small number of very specific situations. Hyperedges with small order represent general information which can be used globally, in many distinct situations [3]. Thus, hypernetworks effectively combine the local and global patterns as an ensemble machine. This mixed-order hypernetworks enhance the efficiency of learning complex behavior from data. Hyperedges of various orders allow the model to find the important patterns, such as short-term and long-term behavior of time series data. Given a sequence of values, the prediction obtained from hypernetwork models provides a unified output, combining the local and global features of the time series that it has learned.

For experiments, two distinct methods of assessing the performance of forecasting models are proposed. By applying the Dow Jones Industrial Average Index and the Korea Composite Stock Price Index data, the experimental results show that the proposed method effectively learns the time series data. In addition, the hypernetwork-based models outperform the popular machine learning methods such as support vector machines, naive Bayes, multilayer perceptrons, and k-nearest neighbors.

The rest of the paper is organized as follows. In Section 2, we present an overview of existing methods for the binary time series problem. Section 3 describes the principles of the hypernetwork model of learning. Section 4 describes the hypernetwork setup for binary time series prediction. In Section 5, the experimental results on the Dow Jones Industrial Average and Korea Composite Stock Price indices are provided. Section 6 concludes the paper and presents directions for future research.

II. Related work

Numerous methods for binary time series analysis are available, mainly in the statistics literature. Most research in this field uses autoregressive models, the Generalized Linear Model (GLM) or Hidden Markov Model (HMM) as starting points, and provides adaptations, variations and generalizations of these techniques to the binary time series problem. Autoregressive and nonparametric additive regressive models for binary time series are considered and some generalizations of these models are proposed by Hyndman [2]. In applications on data from IBM stock transactions and Melbourne's rainfall, he uses lagged values of the response variable in a logistic additive regression and obtaines some

978-1-4244-2959-2/09/$25.00 ©

interpretable results for the forecasting problem. Salas et al. [4] model a binary time series obtained from clipping a continuous valued hydrological series using a periodic discrete autoregressive process. Linear models are applied to the problem of forecasting a binary time series that characterizes the occupancy status of wireless spectrum by Yarkan and Arslan [5]. Binary autoregressive models and Markov processes are applied in Startz [6] to US recession data. A probabilistic methodology based on HMM is used to recognize and predict the sign in short financial trends in Bicego et al. [7]. Unlike other statistical approaches which forecast the transition probabilities or the probability that the returns are above or below a certain threshold value, they use the straightforward modeling of the sign of the returns in short financial trends using HMM.

Artificial intelligence methods, as well as machine learning techniques have been extensively used for the problem of time series forecasting when the data are of continuous nature. Neural networks [8] or evolutionary approaches such as genetic programming [9], [10] have been used and reported very good results. In the case of binary time series, the literature is rather scarce. Recurrent linear networks are used to learn the temporal context associated with a set of binary time series data in Voegtlin [11]. A hybrid system for directional prediction of the stock market is described in Choudhry and Garg [12]. It is based on a genetic algorithm that selects an optimal set of features (from a large set of technical indicators of the market) and feeds it as input to a support vector machine.

Classification models (linear discriminant analysis - LDA, logit) used for the direction of stock index movement outperform the methods that provide level estimation (such as exponential smoothing or multilayered feed-forward neural network) in terms of the trading profit generated by their forecasts, as reported in Leung et al. [13]. The predictability of the direction of stock index movement is examined by means of machine learning methods, specifically classification models, such as LDA, logit, artificial neural network, Random Forest (RF), and Support Vector Machines (SVM) in Kumar and Thenmozhi [14]. The reported empirical results show SVM outperforms the other models, and the RF model is the second best. The performance of SVM for the problem of predicting stock market direction is studied in Juan et al. [15]; the results of comparison with other methods (LDA, quadratic discriminant analysis and Elman backpropagation networks) show that SVM outperforms them.

The binary time series problem may be modeled as a supervised learning problem – in our case, a two class classification problem. Hypernetworks have been successfully applied to solve various machine learning problems [16], [17]. Hence, we examine the potential of applying the hypernetwork model of learning to the task of forecasting a binary time series, and investigate some improvements of the basic model using evolutionary mechanisms. The forecasting performances of the proposed hypernetwork models are obtained in various experimental settings, and are compared with the results of other machine learning techniques.

III. The Hypernetwork Model of Learning

The hypernetwork model is a recently proposed probabilistic graphical model of learning [16], [17], [18], based on random hypergraphs [19]. An edge in a hypergraph is called a hyperedge and it may connect any number of vertices (in usual graphs, an edge connects only two vertices). Hypernetworks are a generalization of hypergraphs, where hyperedges may be of different strengths. The strength of a hyperedge is expressed by assigning it a weight.

For a data set, the hypernetwork has the features of the data set as vertices. Given an input data, hyperedges are constructed by random sampling of the features. The number of features sampled in a hyperedge is called the order of the hyperedge. By means of the hyperedges, a hypernetwork contains partial contents of the input data. A piece of the original information (input data set) can be obtained by unifying a set of hyperedges.

Formally, a hypernetwork is a triple $H = (X, E, W)$, where X is the set of vertices (representing the feature variables in the data set) $X = \{X_1, X_2, \ldots, X_n\}$, E is the set of hyperedges $E = \{E_1, E_2, \ldots, E_m\}$, and W is the set of weights associated to the hyperedges $W = \{w_1, w_2, \ldots, w_m\}$. A hyperedge of order k is constructed for a given input data as a tuple consisting in the values of a randomly chosen set of k feature variables, and the label of the input data, $E_i = (x_{i_1}, x_{i_2}, \ldots, x_{i_k}, y_i)$, $1 \leq k \leq n$, where n is the total number of features in the data. A hypernetwork is said to be k-uniform if every edge $E_i \in E$ has cardinality k. A detailed description of the hypernetwork model of learning is available in [3].

IV. Hypernetwork learning for time series prediction

In this paper, we developed a hypernetwork-based system that can be used to extract patterns from binary time series. In particular, we focus on financial binary time series which are hard to predict due to the complex interactions between players on the market (buyers and sellers) and dependencies on the global and local financial environments. However, many financial market analysts and experienced stock brokers agree that, very often, some patterns emerge from a huge amount of financial data [20]. Identifying these patterns and acting upon them in proper time allows brokers to avoid losses and frequently to obtain high profits. Studies have shown that when many market players use a certain pattern, this pattern loses its profitability and it is replaced quickly by other patterns, newly emerged and more profitable [7]. Very often this replacement is only temporary and the old pattern reemerges on the market when the proper conditions are met.

A. The hypernetwork creation process

The hypernetwork stores significant financial patterns extracted from large amounts of binary time series, and it is used to predict the future evolution of the time series.

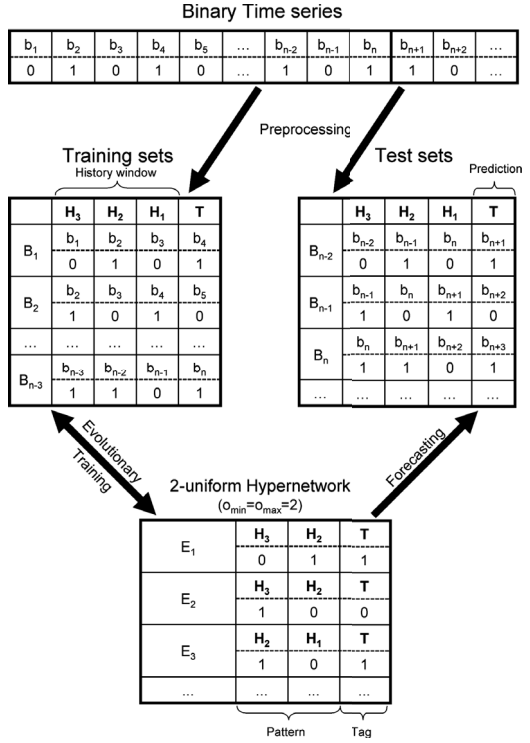

Fig. 1. The procedure of creation and training a hypernetwork on binary time series data, and forecast on the test data

We consider binary time series representing the up/down evolution of stock exchange indices. In this case, the feature variables are the labels for past values in the time series, referred as history window.

Let us consider a generic binary time series dataset $B = \{b_1, b_2, \ldots, b_n\}$, with $b_i \in \{0, 1\}$. In order to create and train a hypernetwork, we first set the size of the history window (m), which controls the number of past values that will be considered when looking for patterns for a certain data point. This means that the hypernetwork will learn patterns from the training sets $B_i = \{b_i, b_{i+1}, \ldots, b_{i+m}\}$ extracted from the original time series for $i = \overline{1, n-m}$. We will refer to the subsets B_i of the original series B as *training sets*, while b_i is merely an observation from the original time series data set.

The process of creating the hypernetwork consists in creating hyperedges from each training set (B_i). The order of each hyperedge is bounded by two parameters of the algorithm, $o_{min}, o_{max} \in \{1, 2, \ldots, m\}$, with $o_{min} \leq o_{max}$. If $o_{min} = o_{max}$, then a fixed-order hypernetwork is created. If $o_{min} < o_{max}$, then a mixed-order hypernetwork is created. In this case, even hyperedges created from the same training set can have different orders.

The number of hyperedges created for each training set (B_i) is also a parameter of the algorithm ($copies$). Usually, this parameter has a fixed value, but a method to compute its value based on features of the training set could also be employed. A hyperedge $E = \{b_{e_1}, b_{e_2}, \ldots, b_{e_k}\}$, where $e_j \in \{i, i+1, \ldots, i+m-1\}$, is created from a training set $B_i = \{b_i, b_{i+1}, \ldots, b_{i+m}\}$ by randomly sampling its k distinct vertices, $k \in [o_{min}, o_{max}]$. The hyperedge is then associated with the tag $t_E = b_{i+m}$ and inserted into the hypernetwork, with the initial weight $w_E = 1$.

In other words, the elements of the hyperedge E represent a pattern extracted from the training data. The hyperedge associates this pattern with the response tag t_E. For example, if $E = b_{i+m-3} = 1, b_{i+m-2} = 1, b_{i+m-1} = 1$ and $t_E = 0$, it can be interpreted as: a decrease will follow after three consecutive increases.

Each training set B_i (subset in the original binary series), coupled with the tag (the next value in the series) represents an actual pattern. The learning process in the hypernetwork model is the step when all the patterns in the time series get stored, in a probabilistic manner, in the hypernetwork. By using hyperedges of orders less than or equal to the actual size of the training sets, the hypernetwork distinguishes among important patterns in the data. This way, given a sequence of values, the prediction obtained from the hypernetwork provides a unified output, since the answer is obtained by merging results from all hyperedges that fit to the sequence.

B. The hypernetwork training process

After creating the initial hypernetwork, an iterative process is used to further train it using the same training sets. The number of iterations over the training sets is a parameter of the training process ($steps$). The training process in the classical hypernetwork model consists in adjusting the weights of the hyperedges, without changing the set of hyperedges. On each iteration j, from each training set B_i a new hyperedge E' of $k \in [o_{min}, o_{max}]$ features is created. Because it was created from the training set B_i, the tag of this hyperedge is $t'_E = b_{i+m}$. The hyperedge E' is then matched against all hyperedges by comparing the features they contain, their values and their tags. The weight of each hyperedge w_E is adjusted for the next iteration $(j+1)$ according to the relation between this hyperedge and the reference (E') with

$$w_E^{j+1} = w_E^j + \delta(E, E'), \tag{1}$$

where δ is a reward function.

The main disadvantage of the training process in the classical hypernetwork model is the high dependency on the sampling procedure employed during the creating process. To overcome this problem, the value of the $copies$ parameter of the algorithm has to be very high, in order to produce hyperedge instances that cover a large area of the search space. The high value of this parameter results in a very large hypernetwork which requires large computational resources in order to be stored and trained.

We propose an evolutionary training process focused on given data, called Data-driven Evolutionary Training (DET). Whenever a training set sample is not matched by any hyperedge, one of the smallest weight hyperedges is selected at random and replaced with a new hyperedge created from the unmatched sample. This approach ensures that the new hyperedge matches the sample, thus adding new information

in the hypernetwork. If the information is valuable, then the weight of the hyperedge increases in time and it will survive. If the information is of little value, then the weight of the hyperedge will remain small and it is possible that the hyperedge will be replaced in the evolutionary training process sometime in the future.

The reward function introduced in (1) is used to compute the variation of the hyperedges weights. Although we call it *reward* function, it can also be used as a penalty function when its results are negative values (e.g. in the case of mismatching tags).

In hypernetworks, the hyperedges of large order represent specialized, thus local, information, while those of small order represent general information, used more globally [3]. Low order hyperedges are more likely to match many training sets, thus having a large number of learning opportunities. Therefore, the variation of such hyperedges must be small in order to produce a smooth learning process. On the other hand, high order hyperedges will match a limited number of very specific training sets. Since there are very few learning opportunities for these hyperedges, we need to make sure that these hyperedges receive a great reward (or penalty) for each matching case.

In order to meet the reward criteria stated above, we define the reward function which rewards matching hyperedges according to their orders. A hyperedge E matches the reference hyperedge E' if E has all the features of E' and these common features have the same value over in both hyperedges. If E matches E' and their tags match, then the reward value is the order of E. If E matches E' and their tags do not match, then the reward value is minus the order of E, which is a penalty. If E does not match E', then there is no reward, but no penalty, too. Formally, this reward function is

$$\delta(E,E') = \begin{cases} k, & \text{if } \forall i \in [1,k'], \exists j \in [1,k] : \\ & \qquad e'_i = e_j \wedge t_E = t'_E \\ -k, & \text{if } \forall i \in [1,k'], \exists j \in [1,k] : \\ & \qquad e'_i = e_j \wedge t_E \neq t'_E \\ 0, & \text{otherwise} \end{cases}, \quad (2)$$

where $E = \{b_{e_1}, b_{e_2}, \ldots, b_{e_k}\}$ is a hyperedge with tag t_E and order k, and $E' = \{b_{e'_1}, b_{e'_2}, \ldots, {}' b_{e'_{k'}}\}$ is the reference hyperedge with tag $t_{E'}$. For fixed order hypernetworks, this reward function takes only two values ($\pm o_{min}$) over all the hyperedges.

When the training process is complete, we can assess the quality of the hypernetwork on the training dataset by iterating through the training sets B_i and counting the number of correct tag answers produced by the hypernetwork, which is later used to compute the hit ratio of the hypernetwork. During this evaluation process, no further training of the hypernetwork takes place.

C. The hypernetwork evaluation process

We use two distinct methodologies for measuring the performance of the hypernetwork on the time series data. In the first case, feedback is provided from each original training set to the hypernetwork, providing the correct tag answer after each training set evaluation. This way, for the next prediction, the correct tag (the one from the original training set) is used in the input data that is presented to the hypernetwork. All predictions are obtained using data from the original data set. This feedback connection results in a step-by-step prediction (SSP) method, which allows the hypernetwork to escape incorrect predictions and possibly restore a correct answering path. Although the hit ratio is improved by the feedback connection, this method limits the prediction horizon of the hypernetwork. The quality of the hypernetwork assessed with feedback connection provides good insight about the quality of the patterns learned by the hypernetwork from the dataset.

The second measuring methodology provides no feedback to the hypernetwork, resulting in an evaluation method based on continuous predictions (CP). In this case, the evaluation process uses an initial training set (consider it like a seed), which is the first training set (B_1) extracted from the original time series dataset. The tag answer provided by the hypernetwork in the case of the seed training set is appended to the training set and will be used for the next evaluation. There is no correction applied to the answer (i.e. no feedback from the original dataset). All predictions, apart from those that use the seed training set, are made using data that has been produced by the hypernetwork. Thus, the input data fed to the network when inquiring for predictions, is composed of the first original values, contained in the training set B_1, the rest being obtained from the predictions already made by the hypernetwork.

The missing feedback connection usually produces lower hit ratios than SSP, because the hypernetwork may stray from the original dataset. However, this method allows us to asses the capability of the hypernetwork to recover on its own from prediction mistakes in the long run. As a prediction mechanism, this method also allows us to make predictions with larger horizon (but less confidence). Hybrid approaches between the two methodologies could also be formulated, and they will be the subject of future research.

D. Time series forecasting

Various parameters of the algorithm control the learning capabilities of the hypernetwork. The size of the history window (m) controls the amount of background information that the hypernetwork can use to learn patterns. The range of the hyperedges orders ($[o_{min}, o_{max}]$) controls the complexity of these patterns. The number of copies ($copies$) and training steps ($steps$) influences the number of these patterns. Increasing these values usually increases the efficiency of the learning process, at the cost of increased requirements of computational power. After the training process is completed, most of these parameters are reflected in the structure of the hypernetwork and can not be changed without breaking it.

However, the size of the history window can be varied within a certain range without this risk. Sometimes, hypernetworks provide higher hit ratios when forecasting data with history windows smaller than the one they were trained with, because they cancel out some very focused patterns which

appeared long time ago in the training set, but are no longer useful. Therefore, we developed a method to estimate the appropriate history window size for forecasting. In order to make f forecasts, a selection dataset is created from the training dataset by selecting the f training sets that are closest, time wise, to the forecasts.

The hit ratio of the hypernetwork is then computed on the selection dataset with any (or both) of the measuring methodologies described in the previous section using history window sizes varying between 1 and m. The window size which produces the best hit ratio is then selected to be used in the forecasting process. In the event of a tie between hit ratios of different window sizes, we usually select the largest windows size, as it allows for more specialized patterns to be used during forecasting.

The forecasting process is similar to the continuous evaluation process described in the previous section, since in real-world problems and off-line forecasting problems we do not know the future evolution of the time series. The step-by-step prediction method can also be used to test the hypernetwork on a test data set or to provide on-line forecasting.

V. Experimental results

Experiments were conducted toward predicting the directions of daily change in some stock price indices. We used in this study three distinct time series which contain data from the Dow Jones Industrial Average (DJI) and the Korea Composite Stock Price Index (KS11). More precisely, the first time series (DJI-93-97) contains daily prices of DJI between February 5^{th}, 1993 and January 20^{th}, 1997, the second time series (DJI-01-08) contains daily prices the DJI between January 2^{nd}, 2001 and October 24^{th}, 2008, and the third series (KS11-97-02) contains daily prices of the Korea Composite Stock Price Index (KOSPI) from July 17^{th}, 1997 to July 1^{st}, 2002. The transformation of the original series into binary time series is straightforward. The 0 encodes the situation when the price in the current day is smaller than the price in the previous day. The 1 encodes a rise in the price with respect to the price in the previous day. The situation when two successive days close with the same daily price is very rare in financial data, and we encode it with 0. For each time series, the last 40 values, representing data for 8 weeks, are used as test set and are not used during the training process. The rest of the data is used to develop the hypernetwork model and to select the window size for forecasting process.

A. Experimental setup

For each time series we trained and compared various hypernetwork models obtained by combining the following features:

- hyperedge orders — in order to test the quality of the hypernetwork models with respect to the complexity of the learned patterns, we used the following values for the bounds of the hyperedges orders: $(o_{min}, o_{max}) \in \{(5,5),(8,8),(11,11),(8,16),(5,16)\}$. The first three pairs of values are used to actually test this dependency, while the last two pairs are used to test the ability of the hypernetwork to properly combine general patterns (small order hyperedges) with very specific patterns (encoded by high order hyperedges).
- evolution — hypernetworks were trained with and without the evolutionary mechanism, allowing us to determine whether it provides any improvements to the classical hypernetwork model.
- order-dependent profit — the reward function from (2) is compared with a classical reward approach which is order-independent:

$$\delta_1(E, E') = sign(\delta(E, E')) \tag{3}$$

Setup name	Order range	Evolution	Reward function
$HN(5,5)\delta$	5-5	no	(2)
$HN(5,5)\delta_1$	5-5	no	(3)
$EHN(5,5)\delta$	5-5	yes	(2)
$EHN(5,5)\delta_1$	5-5	yes	(3)
$HN(8,8)\delta$	8-8	no	(2)
$HN(8,8)\delta_1$	8-8	no	(3)
$EHN(8,8)\delta$	8-8	yes	(2)
$EHN(8,8)\delta_1$	8-8	yes	(3)
$HN(11,11)\delta$	11-11	no	(2)
$HN(11,11)\delta_1$	11-11	no	(3)
$EHN(11,11)\delta$	11-11	yes	(2)
$EHN(11,11)\delta_1$	11-11	yes	(3)
$HN(8,16)\delta$	8-16	no	(2)
$HN(8,16)\delta_1$	8-16	no	(3)
$EHN(8,16)\delta$	8-16	yes	(2)
$EHN(8,16)\delta_1$	8-16	yes	(3)
$HN(5,16)\delta$	5-16	no	(2)
$HN(5,16)\delta_1$	5-16	no	(3)
$EHN(5,16)\delta$	5-16	yes	(2)
$EHN(5,16)\delta_1$	5-16	yes	(3)

TABLE I
CHARACTERISTICS OF THE HYPERNETWORK SETUPS

These variations resulted in 20 types of experimental setups (presented in Table I) to be tested on each of the three datasets. For each setup, 10 different runs were made. The results averaged over the 10 runs are presented in the following.

The common features of all these hypernetwork setups are: the number of copies per training set ($copies = 30$), the number of training iterations ($steps = 30$), and the number of simulations runs per algorithm setup ($runs = 10$). The maximum allowed value for the history window size on each data set is 16.

B. Results

For the mixed-order setups, Table II presents the mean order of the hyperedges in the trained hypernetworks. The results in the section *Arithmetic mean of orders* indicate the mean orders computed when the weights of the hyperedges are ignored. However, hyperedges have weights which influence the importance of their tags during prediction. The values from section *Weighted average of orders* indicate the average orders computed with respect to the weights. By comparing the distribution of orders, we concluded that

Fig. 2. Time series of stock indices used for experiments

Setup name	DJI-93-97	DJI-01-08	KS11-97-02
Arithmetic mean of orders			
$HN(8,16)\delta$	11.99	11.99	12.00
$HN(8,16)\delta_1$	11.99	11.99	12.00
$EHN(8,16)\delta$	11.81	11.21	11.83
$EHN(8,16)\delta_1$	11.81	11.90	11.85
$HN(5,16)\delta$	10.49	10.49	10.50
$HN(5,16)\delta_1$	10.49	10.48	10.50
$EHN(5,16)\delta$	10.34	10.37	10.36
$EHN(5,16)\delta_1$	10.40	10.42	10.42
Weighted average of orders			
$HN(8,16)\delta$	12.02	11.99	12.03
$HN(8,16)\delta_1$	11.54	11.52	11.55
$EHN(8,16)\delta$	11.04	11.21	11.37
$EHN(8,16)\delta_1$	10.98	10.81	10.97
$HN(5,16)\delta$	10.37	10.38	10.55
$HN(5,16)\delta_1$	9.34	9.35	9.54
$EHN(5,16)\delta$	8.81	8.42	8.83
$EHN(5,16)\delta_1$	7.85	7.52	7.91

TABLE II

MEAN ORDER OF HYPEREDGES IN MIXED-ORDER HYPERNETWORKS

Setup name	Training hit ratio	Selection hit ratio	Test hit ratio	History window size
Step-by-step prediction (SSP)				
$EHN(8,16)\delta$	97%	97%	68%	16.0
$EHN(8,8)\delta_1$	91%	96%	67%	16.0
$EHN(5,5)\delta$	59%	67%	65%	15.8
$EHN(5,5)\delta_1$	58%	67%	65%	16.0
$EHN(5,16)\delta$	63%	69%	64%	16.0
Continuous prediction (CP)				
$EHN(5,5)\delta$	56%	63%	63%	16.0
$EHN(5,5)\delta_1$	56%	63%	63%	16.0
$EHN(5,16)\delta_1$	55%	63%	62%	16.0
$HN(5,5)\delta_1$	55%	63%	60%	16.0
$EHN(8,16)\delta$	52%	80%	57%	16.0

TABLE III

PERFORMANCES OF TOP 5 HYPERNETWORK SETUPS ON THE DJI-93-97 DATASET

hyperedges in classical and evolutionary hypernetworks have similar distribution. However, during the training process, evolutionary hypernetworks tend to replace often their high-order low-weight hyperedges. This process is demonstrated by the lower values of the weighted orders. In fact, a similar process takes place in the human brain, which forgets unused and very specific information in order to retain new very specific information. This replacement process has little effect on the important or general information, which in the case of hypernetworks is represented by low-order and high-weighted hyperedges.

Table III presents the results obtained on the first data set by the top 5 best algorithm setups, ordered descending by their prediction accuracy. The upper part of the table contains the mean values obtained for each setup over the 10 simulation runs when using step-by-step prediction. The second column presents the mean hit ratio obtained on the training set after the training process is completed. The third column presents the mean hit ratio obtained on the selection set, which contains 40 observations of the training set closest (time wise) to the test set. Thus, these values represent the performance of the hypernetwork models in the time frame immediately before the time frame of the test set. The fourth column presents the mean hit ratio obtained on the test set, which measures the prediction accuracy of the discovered patterns. In each run, the predictions on the test set are obtained using the history window size (between 1 and 16) which obtained the best results on the selection sets. The mean values of the selected history window sizes for each setup are included in the last column. The lower part of the table contains the same information obtained when continuous prediction is used. The results indicate that the evolutionary hypernetworks perform better than the classical ones, especially with step-by-step prediction. With some exceptions, the hypernetworks used the maximum allowed values for the history size.

One can notice that for continuous prediction, the accuracies obtained on the test dataset are higher than the ones achieved on the training dataset. This is normal behavior considering the time span difference between the two datasets. For training datasets, the hypernetworks have to predict 939 future values (almost 4 years) starting from only 16 original values. The prediction of the future values takes place without any feedback whether the previously predicted values were correct or not. The same principle applies for selection and test datasets, but the lower prediction time span (40 future values) limits the effect of accumulating errors. It should be noted that the worst prediction accuracy (among the 20 setups) was attained in both cases by $HN(5,16)\delta$ (43% for SSP and 19% for CP).

Table IV presents the results obtained on the second data

Setup name	Training hit ratio	Selection hit ratio	Test hit ratio	History window size
Step-by-step prediction (SSP)				
$EHN(8,8)\delta_1$	84%	90%	68%	15.8
$EHN(8,16)\delta$	93%	92%	66%	15.8
$EHN(11,11)\delta_1$	99%	98%	65%	16.0
$HN(8,8)\delta_1$	84%	89%	64%	15.8
$EHN(5,16)\delta_1$	63%	71%	64%	16.0
Continuous prediction (CP)				
$EHN(5,5)\delta$	51%	67%	72%	15.2
$EHN(5,5)\delta_1$	50%	70%	71%	15.4
$EHN(8,16)\delta_1$	51%	87%	67%	16.0
$EHN(11,11)\delta$	51%	98%	61%	16.0
$EHN(8,16)\delta$	51%	87%	59%	15.2

TABLE IV
PERFORMANCES OF TOP 5 HYPERNETWORK SETUPS ON THE DJI-01-08 DATASET

Setup name	Training hit ratio	Selection hit ratio	Test hit ratio	History window size
Step-by-step prediction (SSP)				
$EHN(8,8)\delta_1$	91%	94%	56%	16.0
$EHN(5,16)\delta_1$	71%	75%	54%	16.0
$HN(8,8)\delta$	87%	89%	53%	15.8
$EHN(8,8)\delta$	90%	93%	53%	16.0
$EHN(8,16)\delta$	97%	98%	51%	16.0
Continuous prediction (CP)				
$EHN(8,16)\delta_1$	53%	82%	52%	16.0
$EHN(8,16)\delta$	50%	80%	52%	10.0
$EHN(5,16)\delta_1$	52%	60%	51%	11.8
$HN(5,5)\delta_1$	52%	59%	51%	14.8
$EHN(5,5)\delta_1$	51%	58%	51%	13.4

TABLE V
PERFORMANCES OF TOP 5 HYPERNETWORK SETUPS ON THE KS11-97-02 DATASET

set by the top 5 best algorithm setups. The results are consistent with the observations made on the previous dataset: the evolutionary hypernetworks outperform the classical ones and the history window size is close to the maximum value. Another similar behavior can be noticed with respect to the order domains of the hypernetworks: middle and high order setups, which produce more specific patterns, obtain higher performance with step-by-step prediction, while low order setups, which produce more general patterns, have higher performance with continuous prediction. For this dataset, the worst prediction accuracy (among the 20 setups) was obtained by $HN(5,16)\delta$ (32% for SSP and 30% for CP). Although $HN(5,16)\delta_1$ obtained the same results on the validation dataset, it had slightly better performances on the training and selection dataset.

Table V presents the results obtained on the third data set by the top 5 best algorithm setups. Although the performance attained in this case is lower, the results are consistent with other results reported for time series generated by the KS11 index [21]. One possible explanation for these poor results is the fact that Korea's income gains depend on manufacturing and exports, while US economy depends upon asset inflation in the stock market and real estate to sustain consumer spending. The KOSPI index, which is tracking the level of Korea's exports [22], was affected by Korea's financial crisis which started in 1997 (the starting year of the training dataset), fluctuated until mid-2001, and started to rise rapidly in 2001 (the training dataset ended with April 2002). This phenomenon is visible in Figure 2. Therefore, the training dataset contains a structural change around mid-2001, when patterns learned so far may have become obsolete. When the test data was fed to the hypernetwork, some patterns learned by the model may have been inadequate. In this case, the worst prediction accuracy (among the 20 setups) was obtained by $HN(8,16)\delta$ (33% for SSP and 32% for CP).

In order to properly evaluate the forecasting ability of the proposed models, we used for comparison some machine learning algorithms whose performances are known to be very good [14]. We tested Support Vector Machines with polynomial kernel (SVM), Naive Bayes (NB), Logistic Regression with parameter ridge=10^{-8} (LOGREG), Decision Trees (DT) which implement the $C4.5$ algorithm (referred to as $J48$ in Weka), Multilayer Perceptron (MLP), K Nearest Neighbors with k=4 (KNN), and Random Forests (number of trees=10). The experiments were performed using the default implementations offered by Weka[1]. The results obtained with these algorithms are presented in Table VI. We compared these results with the ones obtained by the hypernetworks with step-by-step prediction, since Weka has only this type of prediction.

For DJI-93-97, all the best five hypernetwork setups (and some other, too) outperformed all other tested algorithms. For DJI-01-08, the best two hypernetwork setups outperform all other algorithms except for MLP, while the other setups provided comparable results with the other machine learning algorithms. A similar situation occurs for KS11-97-02, when the best five hypernetwork setups outperformed all other algorithms, except Random Forest. Overall, the evolutionary hypernetwork model of learning proved to perform very well on the problem of learning and predicting binary time series.

VI. CONCLUSIONS

We proposed an evolutionary hypernetwork model for forecasting price movement direction on stock markets. An evolutionary mechanism is employed in the hypernetwork learning process, which allows the identification of a large number of patterns contained in the time series. The prediction obtained from the hypernetwork combines the patterns learned. Different experimental setups for the hypernetwork were tested on three data sets of stock market indices and the results were compared with state-of-the-art machine learning algorithms. The empirical results suggest that the hypernetwork models obtain very good results and that certain setups of the hypernetwork outperform the machine learning methods we have tested.

The patterns identified by means of the hyperedges can be associated in a way with technical analysis patterns used

[1]Weka is an open-source collection of machine learning algorithms for data mining tasks, available at http://www.cs.waikato.ac.nz/ml/weka/

Algorithm	DJI-93-97	DJI-01-08	KS11-97-02
Training hit ratio			
SVM	49.42%	54.09%	51.97%
NB	55.81%	56.63%	54.04%
LOGREG	55.49%	56.12%	54.21%
DT	82.82%	80.91%	81.06%
MLP	83.03%	71.11%	79.25%
KNN	64.81%	66.49%	64.88%
RF	98.84%	99.17%	99.22%
Test hit ratio			
SVM	47.50%	60.00%	47.50%
NB	57.50%	65.00%	35.00%
LOGREG	57.50%	60.00%	35.00%
DT	60.00%	55.00%	50.00%
MLP	62.50%	75.00%	45.00%
KNN	62.50%	65.00%	47.50%
RF	62.50%	65.00%	62.50%

TABLE VI

PERFORMANCES OF OTHER ALGORITHMS

by financial analysts [20]. Therefore, the final structure of the hypernetwork may offer insights for further analysis of time series data. Future study includes the analysis of patterns identified in the hypernetwork structure, as well as a thorough analysis of the influences that parameters of the model have on the prediction performance.

ACKNOWLEDGMENT

This work was supported by the KRF through Foundational Research Program and the MEST through BK-IT Program. Elena Băutu was supported by CNCSIS under grant TD 199/2007 and together with Andrei Băutu by PNCDI2 under grant NatCOMP 2120/2007. The research reported in this paper was done while Elena Băutu and Andrei Băutu were visiting the Biointelligence Lab at Seoul National University in the autumn of 2008.

REFERENCES

[1] P. F. Christoffersen and F. X. Diebold, "Financial asset returns, direction-of-change forecasting, and volatility dynamics," *Management Science*, vol. 52, no. 8, pp. 1273–1288, 2006.

[2] R. J. Hyndman, "Nonparametric additive regression models for binary time series," in *Proceedings of the 1999 Australasian Meeting of the Econometric Society*, 1999, pp. 7–9.

[3] B.-T. Zhang, "Hypernetworks: A molecular evolutionary architecture for cognitive learning and memory," *IEEE Computational Intelligence Magazine*, vol. 3, no. 3, pp. 49–63, 2008.

[4] J. D. Salas, C.-H. Chung, and A. Cancelliere, "Correlations and crossing rates of periodic-stochastic hydrologic processes," *Journal of Hydrologic Engineering*, vol. 10, no. 4, pp. 278–287, 2005.

[5] S. Yarkan and H. Arslan, "Binary time series approach to spectrum prediction for cognitive radio," in *Proceedings of the 66th IEEE Vehicular Technology Conference*, 2007, pp. 1563–1567.

[6] R. Startz, "Binomial autoregressive moving average models with an application to U.S. recessions," *Journal of Business and Economic Statistics*, vol. 26, no. 1, pp. 1–8, 2008.

[7] M. Bicego, E. Grosso, and E. Otranto, "Recognizing and forecasting the sign of financial local trends using Hidden Markov Models," University of Cagliari and Sassari, Sardinia, Working Paper CRENoS, 2008. [Online]. Available: http://ideas.repec.org/p/cns/cnscwp/200803.html

[8] R. N. Yadav, P. K. Kalra, and J. John, "Time series prediction with single multiplicative neuron model," *Appl. Soft Comput.*, vol. 7, no. 4, pp. 1157–1163, 2007.

[9] W. B. Langdon, "Predicting ten thousand bits from ten thousand inputs," University of Essex, Technical Report CSM-457, 2006. [Online]. Available: http://www.cs.essex.ac.uk/technical-reports/2006/csm457.pdf

[10] M. Santini and A. Tettamanzi, "Genetic programming for financial time series prediction," in *Proceedings of the 4th European Conference on Genetic Programming*, 2001, pp. 361–370.

[11] T. Voegtlin, "Recursive principal components analysis," *Neural Networks*, vol. 18, no. 8, pp. 1051–1063, 2005.

[12] R. Choudhry and K. Garg, "A hybrid machine learning system for stock market forecasting," *Proceedings of World Academy of Science, Engineering and Technology*, vol. 29, pp. 315–318, 2008.

[13] M. T. Leung, H. Daouk, and A.-S. Chen, "Forecasting stock indices: a comparison of classification and level estimation models," *International Journal of Forecasting*, vol. 16, no. 2, pp. 173–190, 2000.

[14] M. Kumar and M. Thenmozhi, "Forecasting stock index movement: A comparison of support vector machines and random forest," Indian Institute of Capital Markets, 9th Capital Markets Conference Paper, 2005. [Online]. Available: http://ssrn.com/abstract=876544

[15] W. Juan, Y. Nakamori, and S.-Y. Wang, "Forecasting stock market movement direction with support vector machine," *Computers & Operation Research*, vol. 32, pp. 2513–2522, 2005.

[16] S. Kim, S.-J. Kim, and B.-T. Zhang, "Evolving hypernetwork classifiers for microRNA expression profile analysis," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2007)*, 2007, pp. 313–319.

[17] C.-H. Park, S.-J. Kim, S. Kim, D.-Y. Cho, and B.-T. Zhang, "Use of evolutionary hypernetworks for mining prostate cancer data," in *Proceedings of the International Symposium on Advanced Intelligent Systems*, 2007, pp. 702–706.

[18] J.-K. Kim and B.-T. Zhang, "Evolving hypernetworks for pattern classification," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2007)*, 2007, pp. 1856–1862.

[19] M. Karonski and T. Luczak, "Random hypergraphs," in *Combinatorics, Paul Erdös is Eighty*, vol. 2, 1996, pp. 283–293.

[20] F.-L. Chung, T.-C. Fu, V. Ng, and R. Luk, "An evolutionary approach to pattern-based time series segmentation," *IEEE Trans. Evolutionary Computation*, vol. 8, no. 5, pp. 471–489, 2004.

[21] K.-J. Kim, "Financial time series forecasting using support vector machines," *Neurocomputing*, vol. 55, pp. 307–319, 2003.

[22] S.-Z. Chiou-Wei and Z. Zhu, "Sources of export fluctuations: empirical evidence from Taiwan and South Korea, 1981-2000," *Journal of Asian Economics*, vol. 13, no. 1, pp. 105–118, 2002.