Predicting S&P500 Index Using Artificial Neural Network

Shanghong Li, Jiayu Zhang, Yan Qi

Predicting S&P500 Index Using Artificial Neural Network

Shanghong Li[1] ,JiayuZhang[2] ,YanQi[1,*]

1 International College Beijing, China Agricultural University, Beijing, China

shanghongli@cau.edu.cn,meganqiyan@sina.cn

2 Continental Capital, Beijing, China

jiayu.zhang@contiasia.com

Abstract. This paper studies artificial neural network algorithm as a means of modelling and forecasting the financial market data. Such method bypasses traditional statistical method to deal with financial time series data. A recurrent neural network model, Elman network, is implemented to incorporate autocorrelation in time series data. A 3-parameter model is chosen to fit and forecast S&P 500 index. The experimental data is from 2000-2007, to screen out the abnormal market environment after 2008 financial crisis.

Key words: S&P 500 index,Elman network,artificial neural network, time series

1 Introduction

The main purpose of this research is trying to build a preliminary base on how to apply the artificial neural network (ANN) method to model and forecast the financial market activities. Peters (1991) state that it's impossible to predict the stock price in future with the historical data and get a payback in return based on the effective market assumption. However, many traditional study and data shows that the markets do not often following the efficient market hypothesis completely. Fama(1977) and Fama (1989)statedcertain financial and economical time series could be used to predict somevariables.Penumadu(1997) showedthat several researchsuch as attempting to interpret complex real-world sensor data, ANN are among those most efficient ways available.Trippi(1996) examined how ANNs could be implemented in financial areas by modeling and predicting the financial time series. It is especially difficult to be modeled by traditional statistical method.

Our work starts from introduction of the fundamental structure of the neural network, till the mostly common used multi-layer model. Then a single-threshold neuron problem as a simple example will be used to illustrate some basic neural network processing features. The special technique of backpropagation and gradient descent algorithm is discussed in detailed to mathematically show how a neural network works. In order to incorporate autocorrelation into time series data, an idea of recurrent networks is introduced to build the time series structure into the neural network internally. As one of the most commonly used recurrent networks, the Elman network is discussed in details as it will be implemented to our real world data.

The S&P 500 index is branded as a bellwether for US economy. It is most widely used for large-cap US stocks and oftenconsidered as a baseline for comparison in stock and mutual fund Performance charts. As a result we choose S&P 500 as our fitting and forecasting target. As a preliminary base, we would like to keep less parameter, so we choose 3 parameters, one for the real estate market (it

* International College Beijing, China Agricultural University, Beijing, China

contributes 35%-40% of the index), one for the yield spread (Huang 1998), and the rest for the short-term interest rate.

It is well known that there was a huge financial crisis and turmoil on the financial market in 2008 as the subprime crisis. In order to save the economy and the financial market from the abyss, the Fed started several rounds of quantitative easing (QE) to support the asset prices. Itis believed that the crisis and the following QE had a huge impact and noise of the financial market time series data and made some data distorted to the external artificial interference, which is not welcomed in the model. Especially the financial data before 2008 was chosen to train and test the model, for the purpose ofavoiding the extreme external noise to impact the model.

2 Artificial Neural Networks

2.1 Overview

ANNs provide anefficientway to approximatevarious kinds of objective functions such as real-valued, discrete-valued and vector-valued. ANNs are amazing because of their capability to imitate some of the human brain's information process, though in a simplistic way. An ANN is a system of neurons organized into a network in which data can be processed like the learning process in human brain.

A neural network is trained to study from the data set by adjusting the values of the weights between neurons. Basically, a specific input leads to a particular target outputin neural networks. When training the networks, the weights are adjusted according to the comparison between the output and the target value, till the error terms fall within a certain error tolerance. In general, a large quantity of data is necessary to obtain a well trainingnetwork. The following figure illustrates the training process.

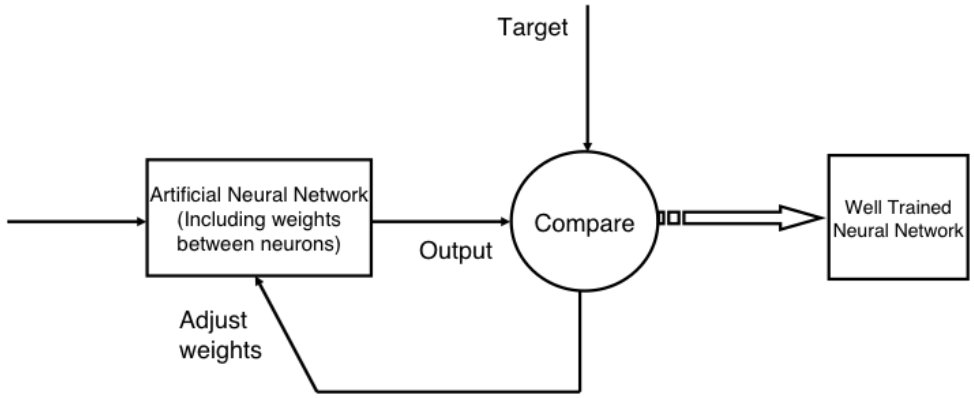


Figure 1:*The process of training neural networks*

Neural networks can perform various tasks, including function approximation, pattern recognition, data clustering and trend prediction. Neural networks have a widely application in areas of business and industry, such as: digital process control, code sequence prediction, voice synthesis, speech recognition, market forecasting, portfolio trading, currency price prediction, and so on.

2.2 Network Structures

Neural networks can have various architectures from simplistic to complicate. The simplest one is the single neuron network which only has one neuron in the network. From simplistic to complicate in structure, we divide neural networks into two groups: the single-layer networks and the multi-layer networks. Then we will give a general description of these typical network structures to give readers an insight to neural networks.

2.2.1 Single-neuron Model

The single-neuron network consists of a scalar input p, a bias input b, a transfer function f and generates an output that is shown in Figure 2.

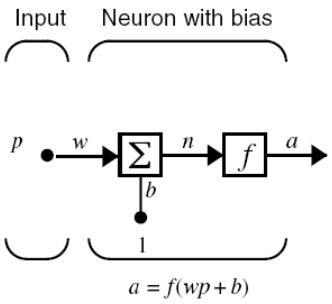


Figure 2:*The single-neuron network structure*

*p*is the scalar inputand *wp*is the weighted input. The transfer function input n is the summation of the weighted input and the bias *b*. *a*isthe output formed by acting the transfer function *f*on the input.Both *w* and *b* are parameters adjustablein the networks. When training the networks the weights and bias parameters will be adjusted based on the training data set to make the networks achieve some desired goal.

2.2.2 Single-layer Model

A single-layer model with *R* inputs and *S* neurons is shown in Figure 3.

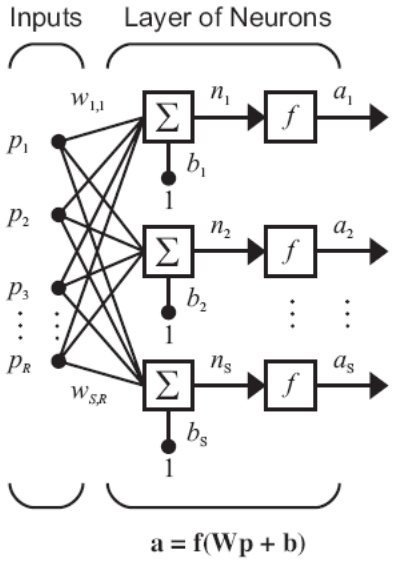


Figure 3:*The single-layer network structure*

In this network, *p*is the input vector and *W*is the weight matrix. $n_i$is a scalar input generated by combine the bias and weighted input of the ithneron.Finally the transfer function *f* acts on the net input *n* to generate a column vector output *a*.In the single-layer network, the input vector enters the network via the weight matrix,

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,R} \\ w_{2,1} & w_{2,2} & \dots & w_{2,R} \\ \vdots & \vdots & \vdots & \vdots \\ w_{S,1} & w_{S,2} & \dots & w_{S,R} \end{bmatrix}$$

The row indices indicate which neuron of the weight, while the column indices indicate which input of the weight. For example, $w_{i,j}$means that weight connected the *i*th input variable to the jth neuron is $w_{i,j}$. Note that the number of input could be different from that of neurons.

The single-layer network with R inputs and S neurons using abbreviated notation, is shown in Figure 4.

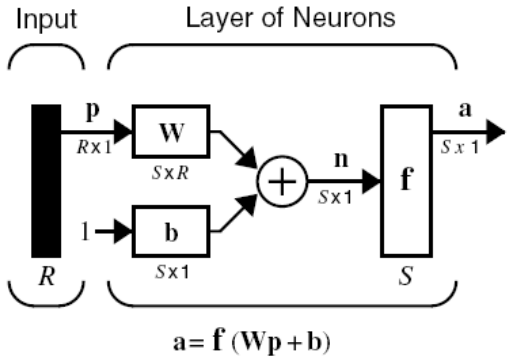


Figure 4:*The single-layer network structure-abbreviated notation*

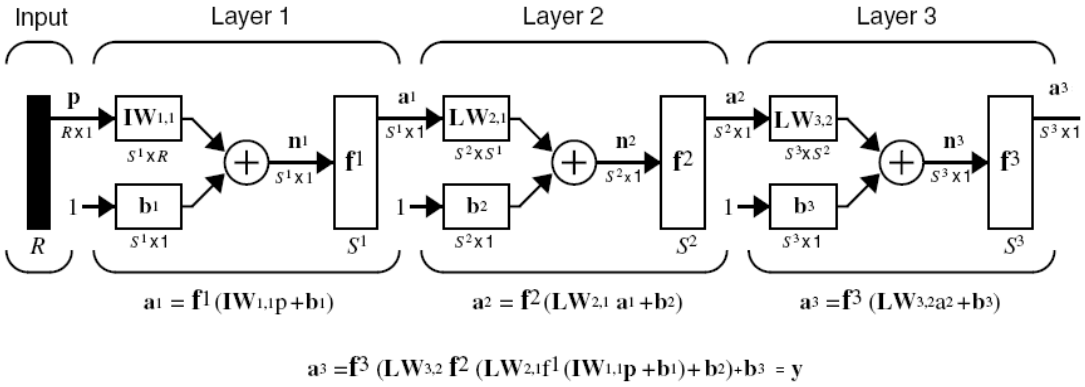
Here *p* is an $R \times 1$ input vector, *W* is an $S \times R$ matrix, and *a*, *b* are $S \times 1$ vectors.

2.2.3 Multi-layer Model

A network could have more than one layer and each layer consist of several neurons. The multi-layer model is most common and powerful neural network structure currently used in the application.

To introduce the multi-layer structure clearly, the notation should be extended. First, The weight matrices connected to inputs are called as input weights(IW), while weight matrices connected between layers are called as layer weights(LW).Secondly, the source(second index) and destination (first index) of the weight matrix should be pointed out. For example, the weight matrix connected the input vector and the first layer is noted as $IW^{1,1}$, which means both the source and destinationare 1.

A three-layer network is shown in Figure 5.



$$a^3 = f^3\,(LW^{3,2}\,f^2\,(LW^{2,1} f^1\,(IW^{1,1}p + b^1) + b^2) + b^3 = y$$

Figure 5:*The three-layer network structure*

The network has R inputs, S^1neurons in the first layer, S^2neurons in the second layer and S^3 neurons in the third layer. The constant input vector 1 represents the bias for each layer.

The outputs of each layer are automatically the inputs for the next layer so the intermediate layer could be treated as a single-layer network. For instance, layer 2 is a single-layer network with S^1 inputs and S^2 neurons and an $S^2 \times S^1$ weight matrix $LW^{2,1}$.$a^1$is the input vector and a^2 is the output vector.

a^3 is the output of the third layer and at the same time is the output of the entire network. The output of the network is always labeled as y. y could be expressed explicitly as:

$$y = f^3 LW^{3,2} f^2 (LW^{2,1} f^1 (IW^{1,1} \cdot p + b_1) + b_2) + b_3)$$

2.3 Network Process

2.3.1 Threshold Neuron Example

We will study a single-threshold neuron problem as a simple example to illustrate some basic neural network processing features.

Table 1:*Classification Data*

X_1	X_2	t
0.2	0.3	0
0.2	0.8	0
0.8	0.2	0
1.0	0.8	1

The problem includes two input variables (x_1 and x_2), one output variable (t) which belongs to one of two categories (0 or 1). The data is given in Table 1. Our task is to correctly divide the input data into two categories (0 or 1). A single-neuron network will be used to achieve this goal, shown in Figure 6.

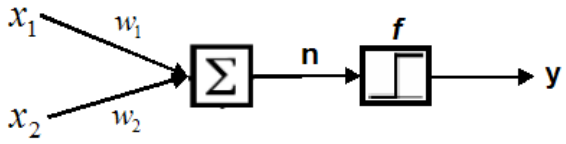


Figure 6:*Liner threshold neuron model*

There are two steps in this problem. First, calculate the net input into the transfer function; second, decide the output using a threshold function. To simplify the problem, the bias is ignored, so when calculating the net input, only the input variables and corresponding weights should be taken into account. In this stage, both of the weights will be fixed at 1 which means that there is no learning.

$$\sum = n = \{w_1, w_2\} \cdot \{x_1, x_2\} = x_1 + x_2$$

So we only need to consider the sum of the two inputs, the threshold should be placed anywhere between 1.0 and 1.8. A threshold will be arbitrarily chosen as 1.3 for this problem. Then the threshold function generates the output ysuch that:

$$f(\Sigma) = y = \begin{cases} 0 & u < 1.3 \\ 1 & u \geq 1.3 \end{cases}$$

Using this simple classifier, it is possible to group the data correctly; the results are shown in Table 2

Table 2:*Performance of the Classifier*

Input(x_1,x_2)	u	y
(0.2,0.3)	0.5	0
(0.2,0.8)	1.0	0
(0.8,0.2)	1.0	0
(1.0,0.8)	1.8	1

From above we notice that a specially designed threshold function could classify the data correctly. However, the drawback of the model is that it does not learn from the environment (weights are equal to 1). Later we will discuss how to train the network to learn from the environment, and consequently how to adjust the weights.

2.3.2 Backpropagation

Standard backpropagation is a gradient descent algorithm, which means that the network weights are adjusted towards the negative direction of the gradient of the network error function. We know that the transfer function is differentiable, and if we define a differentiable error function as the error function of the network, such as the sum-of-squares, then the error function is differentiable with respect to the weights. Therefore we can calculate the derivatives of the errors with respect to the weights anduse the derivatives to adjust the weights so as to minimize the error function by using gradient a descent algorithm or some other more powerful methods.

This technique can be divided into two stages. First, errors are propagated backwards through the network so as to calculate the derivatives with respect to weights. Second, weights are adjusted using calculated derivatives by a certain optimization rule.

We now derive the backpropagation algorithm for a general two-layer feed-forward network. It isstraightforward to generalize the method from the two-layer model to the multi-layer model.For a general feed-forward network, the net input for each neuron is a weighted sum of the inputs of the form

$$n_j = \sum_i w_{ji} p_i \qquad (1)$$

wheren_j is the net input, p_i is the input, w_{ji} is the corresponding weight.

The net input is transformed by a non-linear differentiable transfer function g(·) in the form

$$a_j = g(n_j) \qquad (2)$$

The error function of the network can be expressed as a differentiable function of the network outputs in the form

$$E = E(y_1, \cdots y_n) \qquad (3)$$

Our goal is to find a method for evaluating the derivatives of the error function with respect to the weight. According to Bishop (1996)the partial derivatives is

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial n_j} \frac{\partial n_j}{\partial w_{ji}} \qquad (4)$$

Next, we introduce the useful notation

$$\delta_j = \frac{\partial E}{\partial n_j} \quad (5)$$

Using (1) we can get

$$\frac{\partial n_j}{\partial w_{ji}} = p_i \, (6)$$

Substituting (5) and (6) into (4) we can write

$$\frac{\partial E}{\partial w_{ji}} = \delta_j p_i \quad (7)$$

Thereforeso as to calculate the derivatives, the evaluation of $\delta_j$is necessary for each neuron in the hidden layers and output layer.

For the output layer neuron, the calculation of δ_j is straightforward. From the definition (5) we have

$$\delta_k = \frac{\partial E}{\partial n_k} = g'(n_k)\frac{\partial E}{\partial y_k} \, (8)$$

To calculate δ_j for the hidden layer neurons, we apply the chain rule again,

$$\delta_j = \frac{\partial E}{\partial n_j} = \sum_k \frac{\partial E}{\partial n_k}\frac{\partial n}{\partial n_j} \, (9)$$

where the sum runs over all the output layer neurons k to which the hidden layer neuron j connects.

The structure of neurons and weights is shown in Figure 7.

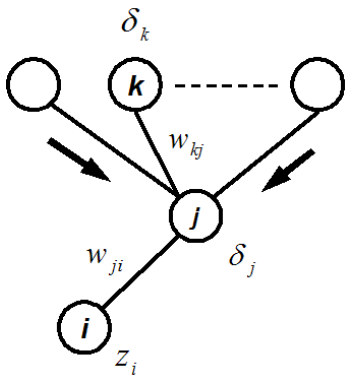


Figure 7:*The evaluation of δ_j for hidden neurons*

For the output layer we have,

$$n_k = \sum_j w_{kj} g(n_j) \, (10)$$

Now substituting the definition of δ_j given by (5) into (9) and use (10), we get

$$\delta_j = g'(n_j)\sum_k w_{kj}\delta_k \, (11)$$

It's clear that the value of δ_j for the hidden neurons could be calculated by backpropagatingδ_k for the neurons higher up in the network.

Thebackpropagation procedure for evaluating the derivatives of the error function is summarized in the following four steps:

- Apply an input data x_n into the network and forward through the layers using (1) and (2) to calculate the output for all neurons in the hidden layers and output layer.
- Calculate$\delta_k$for all output neurons using (8).
- Backpropagate the δ_k using (11) to obtain $\delta_j$for all hidden layer neurons.
- Use (7) to calculate all required derivatives.

2.3.3 Gradient Descent

In order to complete the learning algorithm, we have to provide a method to update the weights based on the weights. The gradient descent technique will be used here.
We know that the error function can be expressed as a differentiable function of weights in the network

$$E = E(w) \quad (12)$$

We begin with an initial guess for w which can be chosen randomly. Then, we update the weights by moving a small step along the direction in which E decreases most rapidly. By iterating this process, a sequence of weights to minimize the error function is generated,

$$w_{ji}^{(\tau+1)} = w_{ji}^{(\tau)} - \eta \frac{\partial E}{\partial w_{ji}}\Big|_{w^{(\tau)}} \quad (13)$$

The choice of η (learning rate) is critical in the algorithm, since if it is too small the reduction of the error will be too slow, while if it is too large, the sequence might diverge.
Substituting (7) into (13), we get

$$\mathrm{V}w_{ji} = -\eta \delta_j p_i \quad (14)$$

3 Time Series Processing

It seems that it's not appropriate to apply neural networks for time seriessince they were built mainly for pattern recognition.Another reason is that the initial applications were dealing withdetection of patterns in arrays of measurement not change in time (Shinichi 2006).Additional structure in the neural network is required for the spatio-temporal property of the time series data.Particularly, memory in time in the structure is necessary for a neural network applied for time series processing.There are two types of nonlinear networks successfully used for time series processing (Box 1965). One is the modified backpropagation network and the other one is recurrent network embedded in the structure to capture the long-term memory. We will focus on the latter one as tool to do the time series processing.

3.1 Recurrent Network

The key idea is to incorporate the autocorrelation property of the time series into the network (Samarasinghe 2006). Such networks couldfigure out the long-term record of the time series by its internal dynamics, instead of depending on external memory filters. Also the recurrent network has feedback loops to memorize the past state of the network and recursively send them back into the network such that the network has an internal long-term-memory property. What's more, memory dynamics and the time lag structure depend on the network itself instead of any other externally designed structure. The way the long-term memory is built into the network is illustrated using a simple model below:

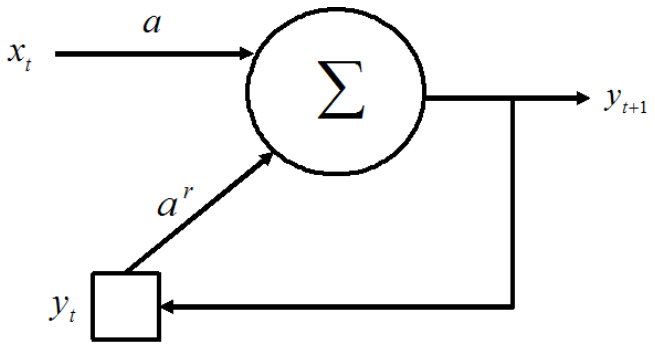


Figure 8:*Recurrent linear neuron*

There are two inputs at time t: external input (x_t), internal delayed input (y_t). The neuron forecasts the next state y_{t+1} and becomes a fresh delayed input stored in the square. The whole process is recursiveand repeated. At time *t*, the output could be expressed as

$$y_{t+1} = a^r y_t + a x_t \quad (15)$$

Where$a^r$is the weight with respect to the delayed input, y_{t+1}then turns into a new delayed input.The output at t+1 is generated in the same manner

$$y_{t+2} = a^r y_{t+1} + a x_{t+1} \quad (16)$$

Substituting (15) into (16), we get

$$y_{t+2} = a^r (a^r y_t + a x_t) + a x_{t+1} = (a^r)^2 y_t + a^{r+1} x_t + a x_{t+1} \quad (17)$$

Suppose that a r =0.2 and a=0.3, the output is

$$y_{t+2} = 0.04 y_t + 0.06 x_t + 0.3 x_{t+1} \quad (18)$$

We can see that the network has built in the last two external input and the last predicted output into the structure. This built in structure enables the model to have a long-term memory capability. The recurrent weight determines the weight put on the time lags. Recursively

$$y_{t+n} = (a^r)^n y_t + (a^r)^{n-1} a x_t + (a^r)^{n-2} a x_{t+1} + \cdots + a x_{t+n-1} \quad (19)$$

The weight will be adjusted during the process.When the nonlinear transfer function f is embedded into the model (take time step t+2 as an example), we get

$$y_{t+n} = f\left[a^r f(a^r y_t + a x_t) + a x_{t+1} \right] \quad (20)$$

3.2 Elman Networks

Elman and Jordan networks are the two most popular recurrent networks in application. Elman network, mostly a two-layer network, is used in our application and discussed in details here. It's shown as below with feedback loop from the first layer output to the first layer input.

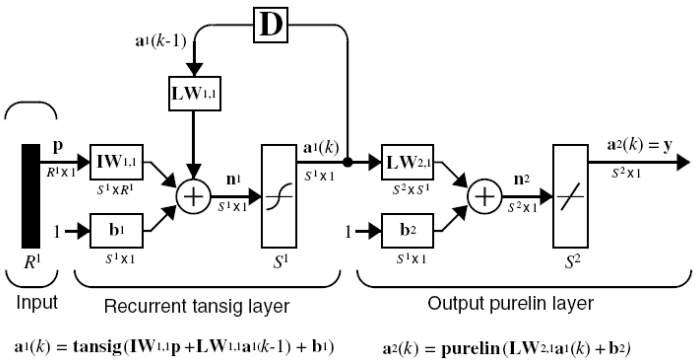


Figure 9:*Two-layer Elman network*

Elman network contains both*tansig* transfer function and *purelin*transfer function. Therefore it's possible to approximate arbitrary function with any accuracy.To show how the Elman network works,a small network will be studied (the bias is ignored for simplicity).

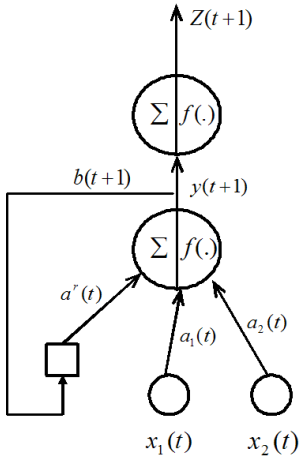


Figure 10:*Simple Elman recurrent network*

Here, x_1 and $x_2$are two given inputs in the network andy is an internal delayed input. The delayed input at every previous step needs to be storedin a unit and the output resultfrom the hidden layer will be fed back in the next time step.$a_1(t)$, $a_2(t)$ are hidden layer weights and $a^r(t)$ ($a^r(t)$ denotes the recurrent weight). The process from the hidden layer and up is defined as time step *t+1*. So the hidden layer output weight is b(t+1) .

Foreach time *t*, the weighted input u(t+1) and output y(t+1) for the hidden layer are given by

$$u(t+1) = a_1(t)x_1(t) + a_2(t)x_2(t) + a^r(t)y(t) \quad (21)$$

$$y(t+1) = f(u(t+1)) \quad (22)$$

Next, y(t+1) is stored in a unit and will be send back as a delayed input. For the output layer, the weighted input v(t+1) and output z(t+1) are given by

$$v(t+1) = b(t+1)y(t+1) \quad (23)$$

$$z(t+1) = f(v(t+1)) \quad (24)$$

where z(t+1) is the predicted value of the target value T(t+1) . If the error function is a standard sum-of-square function, then the error for the current step is

$$E(t+1) = \frac{1}{2}[z(t+1) - T(t+1)]^2 \quad (25)$$

Then the backpropagation algorithm, introduced in 2.3, will be used to update the regular weights as well as the recurrent weight. The details won't be expanded on here.

4 S&P 500 IndexForecasting

The S&P 500 is a widely monitored and benchmarked U.S. equity market index, which is consistent of the stocks of top 500 corporations with large market-cap, most of which are based in U.S. All component stocks in the index are large and publicly held companies, which are tradedin the top two US equity markets, the NYSE& NASDAQ. As important as the Dow Jones Industrial Average, the S&P 500 is widelyreferred as the representative index for the US large-cap stocks. As the performance metrics for stocks, equity mutual fund and hedge fund, the S&P 500 index is often chosen as a baseline for performance comparison.

Investors use the index to track the up and down of the U.S equity market and besides that they also trade index based financial products, such as the futures and options for the S&P 500 index. So the future trend of the S&P 500 index is not only the indicator of the future stock market, but also could help the investor make an investment decision and hedge against risk.The historical data of the index can be viewed as time series data. Our task is to try to use appropriate economics indices with the help of the artificial neural network to forecast the future trend of the S&P 500 index several months ahead.

4.1 Input Variables

The appropriate choice of the input variables seriously determines the efficiency and the performance of the model. After consideration and filtering, several economical data become our candidate variables. They are new housing starts, mortgage rates, producer price index, consumer price index, Moody's AAA corporate bond index, Moody's BAA corporate bonds index, 3-month commercial paper and energy price. After testing, the combination of new housing starts, Moody's AAA corporate bond index and 3-month commercial paper gives the best performance among those variables. The underlying explanation of choosing theses variables will be discussed below.

4.1.1 New Housing Starts

The S&P 500 index is a market weighted index, each stock's weight in the index is proportional to its market value. Table 3 illustrates the current top ten sectors contributed to the index

Table 3:*Top ten sectors in S&P 500 index*

Sector	% of Index
Financial Services	20.3
Healthcare	13.4
Industrial Material	12.2
Hardware	10.8
Consumer Goods	9.7
Consumer Services	8.8
Energy	6.5

Software	4.5
Business Services	3.9
Media	3.9

The real estate market plays a significant role in the US economy and drives the economy heavily. From the recent mortgage crisis we notice that if the real estate market slumps, the US economy as well as the stock market will be hurt heavily. On the other hand, if the real estate market booms, it will drives the demand for industrial materials, consumer goods and energy consumption. The modern finance industry, closely related with the real estate market, will thrive at the same time.

From Table 3, we notice that these sectors contribute to 35%-40% of the index. We pick the new housing starts as an indicator of the real estate market. Since the new housing starts should have a lagging effect on the real estate market and the whole economy, we have good reason to assume that the S&P 500 index will move up over a certain time lag if the new housing starts increases.

Since the above sectors only contribute to 35%-40% of the index, only the new housing starts will not be enough to capture the future trend of the S&P 500 index accurately if no other variables introduced into the model. Two other variables are introduced in (4.1.2) and (4.1.3).

4.1.2 Moody's AAA Corporate Bond

Moody's AAA Corporate Bond, short for "Moody's AAA", is the index of investment grade bond given an AAA rating by Moody's. It is often viewed in macroeconomics as an alternative to the Fed10-year Treasury Bill as an interest rate indicator.

One parameter that has been proved very effective in predicting the US economy real growth is the interest rate difference between the long-term and short-term debt (*yield spread*). So we pick the Moody's AAA as an alternative indicator to the long-term interest rate.

4.1.3 Commercial Paper

Commercial paper is a kind money-market security issued by largecompanies or banks. Basically,it use to purchase inventory or to manage working capital instead of long-term financialinvestments.

We pick the 3-month commercial paper as an alternative indicator to the short-term interest rate into the model.

4.2 Experiment Result

We use monthly historical data to train our model and make a prediction. We pick the Elman network as the network structure to process the time series data. Since it is a nonlinear dynamics network, there is just an output but no explicit function.

100-month historical data is used to train the Elman network before each prediction and then a 10-month ahead prediction is made based on the trained network.

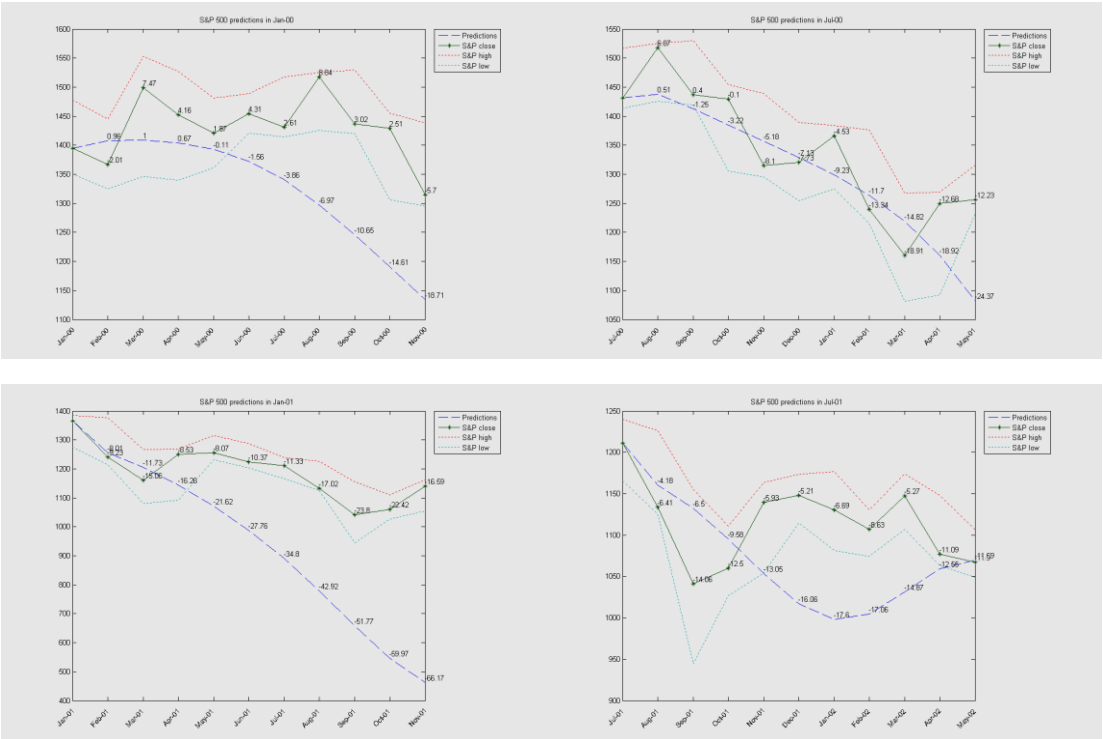
The input variables are sequences of data consisting of the Moody's AAA corporate bond, 3-month commercial paper and the new housing starts in 6 months ago. The target variable is of course the S&P 500 index. Before entering the network, variables have been preprocessed by subtracting the mean and passing the data through a filter to get rid of the noise.

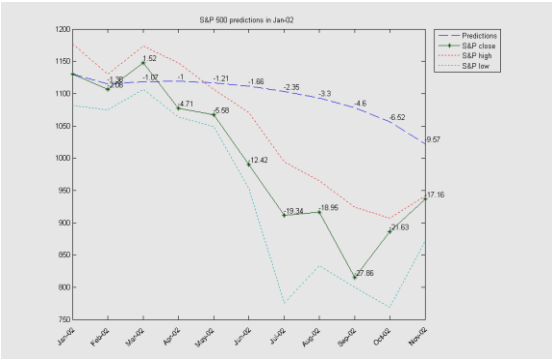
The Elman network we build is a two-layer network with 8 neurons in the hidden layer and the output layer with single neuron. The tan-sigmoid transfer function is used in the hidden layer and the purelin

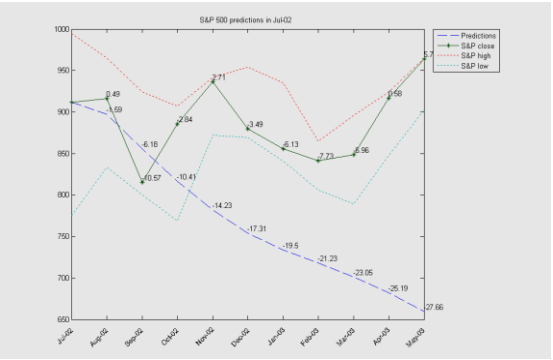
transfer function is used in the output layer. The sum-of-squares is used as the error function. MATLAB R2007a is the working environment. The *trainbr* is picked as the training function.
The simulation runs from Jan-2000 to Jan-2007 semiannually. The following figures show how the model predicts the 10-month ahead trend of the S&P 500 index. The values on the curve show the ratio of change in index from the beginning date.

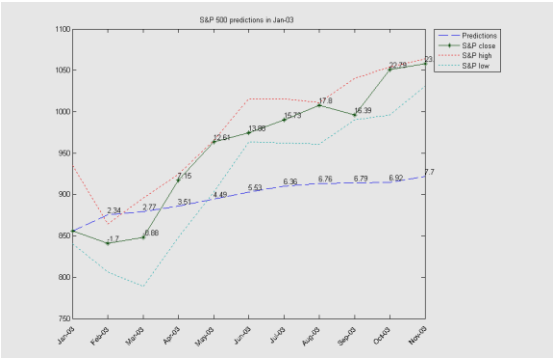
5 Conclusions

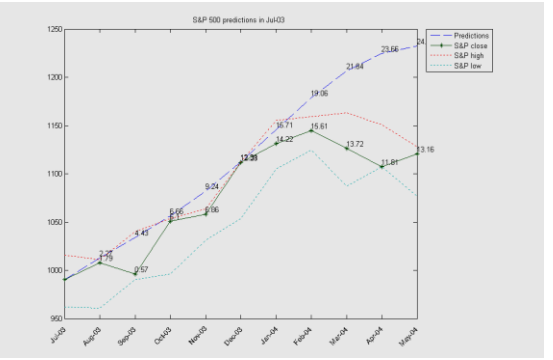
The result shows that with the help of the neural network the new housing starts and long-term, short-term corporate bond could be applied to predict the future trend of the S&P 500 index successfully. When the new housing starts increasing, which means the boom of the real estate market, the whole economy of US will be driven up. S&P 500 index consists of leading companies from widely ranged economics sectors, so the index will go up. On the hand, the index will decrease if the real estate market slumps.
Though we get a good result of the model, there is still a lot to be improved. We need to make our model more robust to contain more parameters and could be more effective for the financial market data after 2008, when there are more noises in the market.
Another problem is that for the nonlinear neural network, there is not an explicit function between the input and output variables. Besides that, the network is not unique either because of its non linearity. It is challenging for us to judge and pick a best model among trained networks.

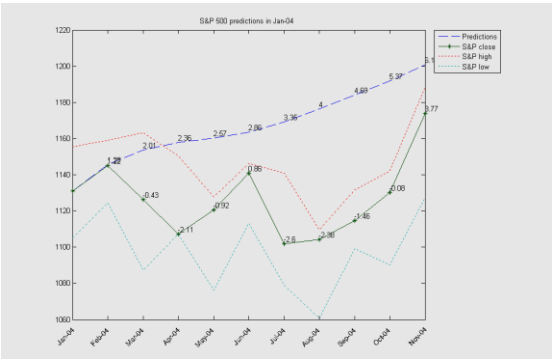


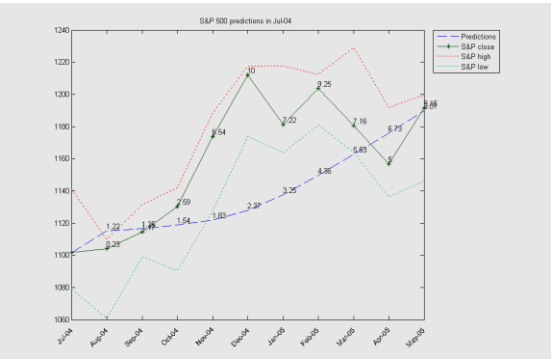

S&P 500 predictions in Jan-02
Predictions
S&P close
S&P high
S&P low

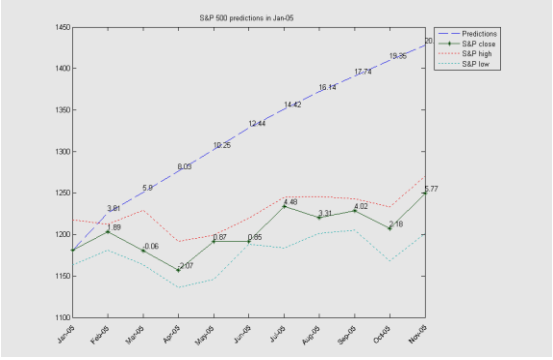

S&P 500 predictions in Jul-02
Predictions
S&P close
S&P high
S&P low

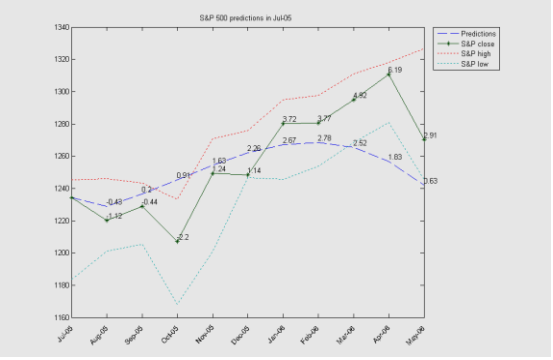

S&P 500 predictions in Jan-03
Predictions
S&P close
S&P high
S&P low

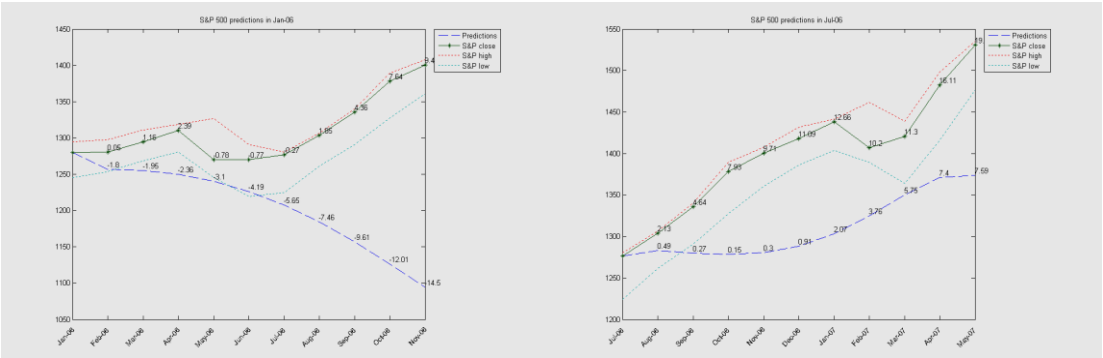

S&P 500 predictions in Jul-03
Predictions
S&P close
S&P high
S&P low


S&P 500 predictions in Jan-04
Predictions
S&P close
S&P high
S&P low


S&P 500 predictions in Jul-04
Predictions
S&P close
S&P high
S&P low


S&P 500 predictions in Jan-05
Predictions
S&P close
S&P high
S&P low


S&P 500 predictions in Jul-05
Predictions
S&P close
S&P high
S&P low



References

[1] Bishop, Christopher. Neural networks: A pattern recognition perspective. *Handbook of neural computation,* 1996.

[2] Box, G.E.P, G.C. Tiao. A change in level of a non-stationary time series.*Biometrica,* 1965, 53:23-35

[3] Fama EF, French KR. Business conditions and expected returns on stocks and bonds. *J Financ Econ,* 1989, 25:23-49.

[4] Fama EF, Schwert WG. Assert returns and inflation. *J Financial Econ*, 1977, 5: 115-146.

[5] Hornick K, Stinchcombe M, White H. Multilayer feed forward network are universal approximate. *Neural Net,* 1989, 2:359-366.

[6] Huang, R.D.,H.R. Stoll. Is it time to split the S&P 500 future contract.*Financial Analyst Journal,* 1998, 52:181-192.

[7] Penumadu, Dayakar, and J. L. Chameau. Geomaterial modeling using artificial neural networks. Artificial neural networks for civil engineers: fundamentals and applications, N. Kartam, I. Flood, and J. Garrett, eds., ASCE, 1997: 160-184.

[8] Peters EE. Chaos and order in capital markets: a new view of cycles, price, and market volatility. Wiley, New York. 1991.

[9] Samarasinghe.Neural networks for time series forecasting.Neural Networks for applied sciences and engineering from fundamentals to complex pattern recognition. 2006.

[10] Shinichi Oeda. Time series data classification Using recurrent neural network with ensemble learning. Lecture notes in computer science. 2006.

[11] Trippi RR, Turban E. Neural networks in finance and investing: using artificial intelligence to improve real-word performance. *Irwin Professional Pub, Burr Ridge,* 1996.