THE KOREAN ECONOMIC REVIEW
Volume 18, Number 1, Summer 2002

FINANCIAL FORECASTING THROUGH DATA MINING: A COMPARATIVE EVALUATION OF PROBABILISTIC NEURAL NETWORKS AND OTHER MODELS

SE-HAK CHUN* · STEVEN H. KIM**

In recent years, neural networks have been applied extensively to the task of predicting financial variables. Even among neural network techniques, backpropagation algorithm has been the most popular methodology. However, the advantages of other learning techniques such as the swift response of the probabilistic neural network (PNN) suggests the desirability of adapting other models to the predictive function. Unfortunately, the conventional architecture for probabilistic neural networks yields only a bipolar output corresponding to Yes or No; Up or Down.

This limitation may be circumvented in part by using a graded forecast of multiple discrete values. More specifically, the approach involves a bipolar architecture comprising an array of elementary PNNs. This paper explores a number of interrelated topics: (1) presentation of a new architecture for graded forecasting using an arrayed probabilistic neural network (APN); (2) use of a "mistake chart" to compare the accuracy of learning systems against default performance based on a constant prediction; and (3) evaluation of several backpropagation models against a recursive neural network (RNN) as well as PNN, APN, and case based reasoning. These concepts are investigated against the backdrop of a practical application involving the prediction of a stock market index.

JEL Classification: C1, C8
Keywords: Data mining, Backpropagation neural network, Recurrent neural network, Probabilistic neural network, Case based reasoning

I. PURPOSE

A systematic approach to knowledge discovery for stock market prediction

Received for publication: Oct. 10, 2001. *Revision accepted:* May 22, 2002.

* *Korea Advanced Institute of Science and Technology 207-43 Cheongryangri, Dongdaemoon-gu Seoul 130-012, Korea* Corresponding author. Tel.: +82-2-958-3686; Fax: +82-2-958-3604; E-mail:chun1120@kgsm.kaist.ac.kr

** *Sookmyung Women's University 53-12 Chungpa-dong 2 Ka, Yongsan-ku, Seoul, 140-742, Korea*

must be able to accommodate disparate types of information. To this end, a battery of techniques from the field of data mining may be harnessed to the predictive task. A key advantage of a multistrategy approach to discovery and forecasting lies in the ability to merge data available in diverse formats.

To an increasing extent over the past decade, software learning methods including neural networks have been used for prediction in financial markets and other areas. By far the most popular type of neural network has been backpropagation. However, the advantages of other learning techniques such as the swift response of the probabilistic neural network (PNN) suggests the desirability of adapting other models to the predictive function. Unfortunately, the conventional architecture for probabilistic neural networks yields only a bipolar output corresponding to *Yes* or *No*; *Up* or *Down*.

This limitation may be circumvented in part by using a graded forecast of multiple discrete values. More specifically, the approach involves a bipolar architecture comprising an array of elementary PNNs. This paper explores a number of interrelated topics: (1) presentation of a new architecture for graded forecasting using an arrayed probabilistic neural network (APN); (2) use of a "mistake chart" to compare the accuracy of learning systems against default performance based on a constant prediction; and (3) evaluation of several backpropagation models against a recursive neural network (RNN) as well as PNN, APN, and case based reasoning. These concepts are investigated against the backdrop of a practical application involving the prediction of a stock market index.

II. BACKGROUND

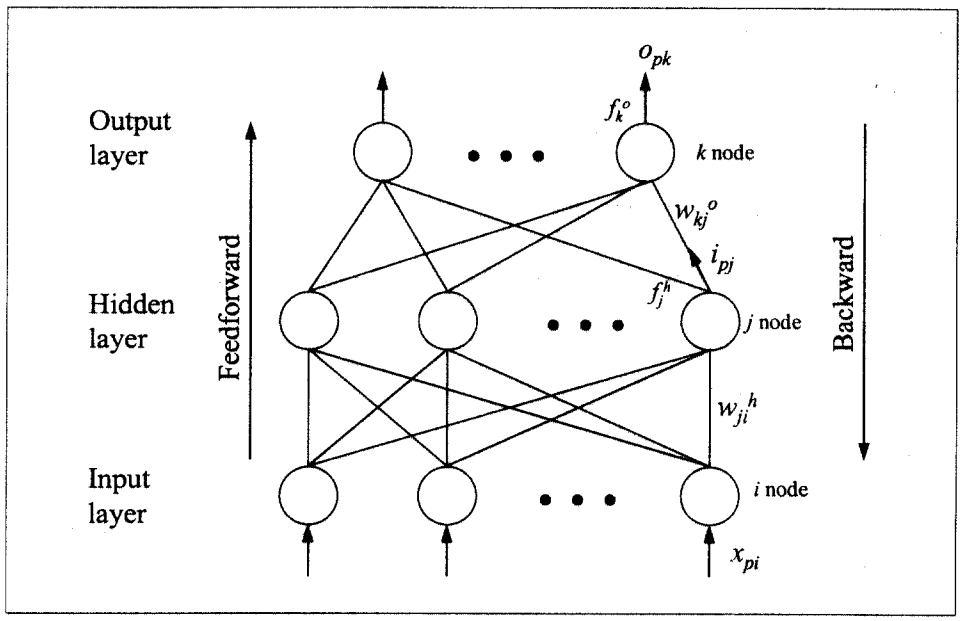
A versatile approach to self-organization lies in neural networks (Anderson and Rosenfeld, 1988; Grossberg, 1976; Hebb, 1949; Hopfield, 1982; Kohonen, 1984; Rosenblatt, 1962; Rumelhart et al., 1986). Neural networks are characterized by learning capability, the ability to improve performance over time. A closely related feature is that of generalization, relating to the recognition of new objects which are similar but not identical to previous ones. An additional characteristic relates to graceful degradation: the network fails gradually rather than catastrophically when it suffers partial damage.

Backpropagation neural network. The neural network methodology has been applied extensively to solve practical problems following the publication of the backpropagation algorithm for the multilayer perceptron (Rumelhart, 1986). The algorithm was developed for the perceptron model, a simple structure to simulate a neuron (Rosenblatt, 1957). Today the backpropagation network (BPN) is the most widely used neural algorithm in science, engineering, finance and other fields.

The general structure of a multilayer perceptron plus the backpropagation

algorithm is depicted in Figure 1. The data entering an input node is multiplied by a set of weights. All such weighted inputs are summed at each node of next layer. The summed value enters an activation function which depends on the learning algorithm. The output of the activation function then becomes the raw input for a node in next layer. This process is called *feed-forward.*

[Figure 1] General structure of the multilayer perceptron using the backpropagation algorithm.



The output of nodes in the last layer may differ from the target value because the weights are initialized randomly. The error between the target value and the calculated value can be adjusted by varying the weights. The weights are adjusted by a delta rule derived from a cost function which depends on the error. This process propagates backward from the output layer to the input layer. The backpropagation algorithm is summarized in Figure 2.

The BPN has several disadvantages. Since BPN relies on a *gradient descent method*, it may slip into a local minimum. In that case the optimum model may be unattainable without using some other appropriate technique. Taking a long time to learn a model is also one of the disadvantages of BPN.

Probabilistic neural network. The probabilistic neural network (PNN) operates on data belonging to a specified number of output categories. Unlike backpropagation networks, a PNN requires only a single presentation of each pattern. The pseudocode for training and utilizing the model is listed in Figure 3.

[Figure 2] The backpropagation algorithm.

1. Enter input patterns, $X_p = (x_{p1}, x_{p2}, \ldots, x_{pN})$ to input nodes (N).
2. Evaluate the net-input to hidden nodes (L):
   $$net_{pj}^{h} = \sum_{i=1}^{N} w_{ji}^{h} x_{pi} + \theta_j^h$$
3. Evaluate the output of hidden nodes:
   $$i_{pj} = f_j^h(net_{pj}^{h})$$
4. Evaluate the net-input to output nodes (M). This process is similar to step 2.
   $$net_{pk}^{o} = \sum_{i=1}^{L} w_{kj}^{o} i_{pj} + \theta_k^o$$
5. Evaluate the output of output nodes:
   $$o_{pk} = f_k^o(net_{pk}^{o})$$
6. Evaluate the errors between the target values and the evaluated output.
   $$\delta_{pk}^{o} = (y_{pk} - o_{pk}) f_k^{o'}(net_{pk}^{o})$$
7. Evaluate the errors for the hidden nodes from the above evaluation (Step 6).
   $$\delta_{pj}^{o} = f_k^{h'}(net_{pj}^{h}) \sum_{k=1}^{M} \delta_{pk}^{o} w_{kj}^{o}$$
8. Revise the weights to the output nodes:
   $$w_{kj}^{o}(t+1) = w_{kj}^{o}(t) + \eta \delta_{pk}^{o} i_{pj}$$
9. Revise the weights to the hidden nodes:
   $$w_{ji}^{h}(t+1) = w_{ji}^{h}(t) + \eta \delta_{pj}^{h} x_i$$
10. Evaluate the cost function. Learning is continued until E_p is under the predefined criteria.
    $$E_p = \frac{1}{2} \sum_{k=1}^{M} \delta_{pk}^{o\,2}$$

A PNN utilizes an exponential function in lieu of the sigmoid activation function often used in neural networks. With this substitution, a PNN can identify nonlinear decision boundaries which approach the Bayes optimal (Sprecht, 1990). PNN appears to outperform backpropagation in discovering local patterns in a time series, especially in the absence of noise.

Recurrent neural network. The recurrent neural network (RNN) consists of two main parts. The forward component has a feedforward data flow pattern which is trained using the backpropagation algorithm. The other module of an RNN is the recurrent component. This module involves a context vector which stores a single period delay of the state of the hidden layer. In particular, the activations in the cells of the hidden layer at time t - 1 are copied into the context vector which re-supplies the hidden layer at time t.

[Figure 3] Pseudocode for training and testing a probabilistic neural network. All wording after the double slash ("//") represents explanatory remarks.

```
Procedure TrainPNN{
    preprocess data by normalizing.
        // or standardize data to simplify choice of s (smoothness parameter)
        // in EmployPNN procedure.
    read data from preprocessed training file.
    for each pattern X,
      create a separate pattern unit
      and set the weight vector Wi equal to X.
                connect the pattern unit's output to the appropriate
      (positive or negative) summation unit.
}

Procedure TestPNN{
    preprocess test data by normalizing; // or by standardizing, as in TrainPNN.
    read test data;
    for each test pattern,
      for each unit in the pattern layer. // di = || Wi − X ||.
      get the distance di between X and the weight vector Wi. // di = || Wi − X ||.
      transform each distance di by proper exponentiation; // exp(- di^2/(2* s^2)).
      sum the distances di;
      connect each pattern unit's output to its corresponding (positive or negative)
      multiply the output from negative summation unit by C = - (n1 / n-1) *
      (h-1 L-1)/(h1 L1)
          // subscript k = 1 for positive cases and -1 for negative cases.
          // hk = prior probability of k; Lk= loss due to action k; nk
              = observation of k.
          // if prior probabilities are unknown, choose C = 1.
      sum up values from the summation units;
      if sum is positive, return 1.
                    otherwise, return -1.
}
```

The computational consequence is a recurrent, fully connected flow of data among all the nodes in the hidden layer. A recurrent network may therefore respond differently to the same input at different times, depending on the prior state of the hidden layer.

Case based reasoning and composite neighbors. Conventional methods of

prediction based on discrete logic usually seek the single best instance, or a weighted combination of a small number of neighbors in the observational space. For instance, the rule of thumb in case based reasoning (CBR) is to seek the nearest neighbor to a target case. In an analogous way, certain algorithms in neural networks seek a fixed number of the closest neighbors; this approach is illustrated by the use of self-organizing maps for pattern recognition tasks (Kohonen, 1984).

An intelligent learning algorithm should therefore take account of a "virtual" or composite neighbor whose parameters are defined by some weighted combination of actual neighbors in the case base. In this way, the algorithm can utilize the knowledge reflected in a larger subset of the case base than the immediate collection of proximal neighbors. The procedure for case reasoning using composite neighbors is presented in Figure 4.

[Figure 4] Predictive procedure through case reasoning using composite neighbors.

Step 1. Begin with current case $x(t)$.

Step 2. Seek the J neighboring cases $x(t_i)$ in the past which are closest to $x(t)$ according to the distance function :

$d_i \equiv d[x(t_i), x(t)]$

Step 3. Compute the sum of weights :

$$d_{TOT} = \sum_{i=1}^{J} d_i$$

Step 4. Determine the relative weight of i^{th} neighbor :

$$w_i = \frac{1}{J-1}\left[1 - \frac{d_i}{d_{TOT}}\right]$$

Step 5. Find the successor $x(t_i+1)$ of each case $x(t_i)$ in the set of neighbors.

Step 6. Calculate the forecast for $t+1$ as the weighted sum of successors:

$$\hat{x}(t+1) = \sum_{i=1}^{J} w_i x(t_i+1)$$

The key to the composite approach lies in the determination of the most effective set of weights to use in order to construct the virtual neighbor. Learning the optimal set of weights is the primary challenge, and the particular values of the weights may well evolve over time as the experience base expands. A promising way to address this task lies in simulated annealing: the weights for constructing the composite neighbor may be perturbed randomly and the advantageous trends pursued, as in the quest for effective parameters in a neural network algorithm.

III. METHODOLOGY

A probabilistic neural network offers swift response, but presents a liability due to its limited output which corresponds to only two states. This limitation may be partly overcome by constructing a battery of probabilistic neural networks. The composite configuration may then be harnessed to provide a graded forecast from the set of bipolar outputs generated by the component networks. The architecture for the arrayed probabilistic network (APN) is presented in Figure 5.

[Figure 5] An arrayed probabilistic neural network (APN). In the APN, the Decision Layer combines the outputs from the Array layer and generates an output. In the current study the Decision layer relies on a positive output, if any, from the elementary PNN at the highest Level in the Array layer. For instance, if Level 3 is the highest level producing an output of "1", the Decision layer declares an overall output of Level 3: this corresponds to a [-0.1, +0.1) percent change in the target variable.

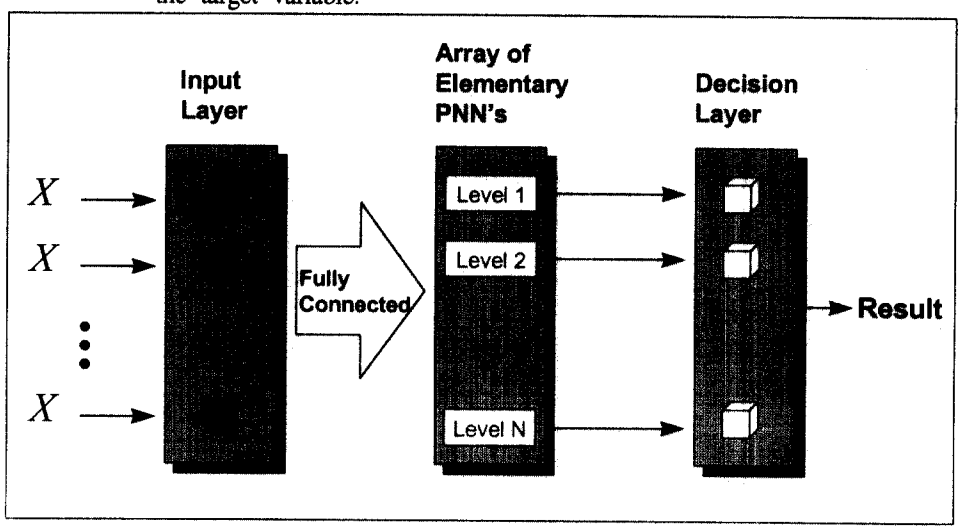


Figure 6 shows the overall procedure for this study. After an exploratory data analysis, the data are converted into a quasi-stationary series through logarithmic and differencing operators where appropriate. Next, a standardization ensures that the values of input variables are of similar order of magnitude. Subsequently, a discrete output is generated. The quantized output is a natural consequence of the APN.

On the other hand, techniques such as RNN yield continuous outputs. In that case, the output values may be discretized to ensure comparability with those

from an APN. Finally, the output from the APN is compared against those from other learning techniques: BPN, RNN, and CBR.

[Figure 6] Procedure for predicting 6 Levels of bipolar movements (*Up* or *Down*) for a stock index using probabilistic neural networks and other learning methods.

1. Perform exploratory data analysis (EDA): identify overall patterns and outliers.

2. Transform data for comparability.

   a. Convert indices (e.g. Stock Price Index) and ratio (e.g. PI) by logarithmic mapping:
   $X_{ij} \rightarrow LX_{ij}$: for variable $i = 1..I$ and case $j = 1..J$

   b. Take differences to eliminate trend if appropriate
   $U_{ij} \rightarrow DU_{ij}$

   c. Standardize: eliminate effects of units (of measurement) by subtracting mean and dividing by standard deviation:
   $V_{ij} \rightarrow ZV_{ij} \equiv Z_{ij}$

3. Generate a multilevel classification of output patterns based on actual stock price index.

   → 6 classes of output with the following thresholds:
   (-0.5%, -0.3%, -0.1%, 0.1%, 0.3%, 0.5%).
   → Each threshold defines two output nodes (e.g. less than 0.1% or 0.1% or more).

4. Apply probabilistic neural network (PNN) or other model.
   → Calculate result from the model.

5. Detransform the variables.
   → Analyze results.

IV. CASE STUDY

The case study involves a bipolar prediction of *Up* or *Down* for the Singapore stock price index. The variables of interest for this study are listed in Table 1. The input data were daily values. The learning phase consisted of 2870

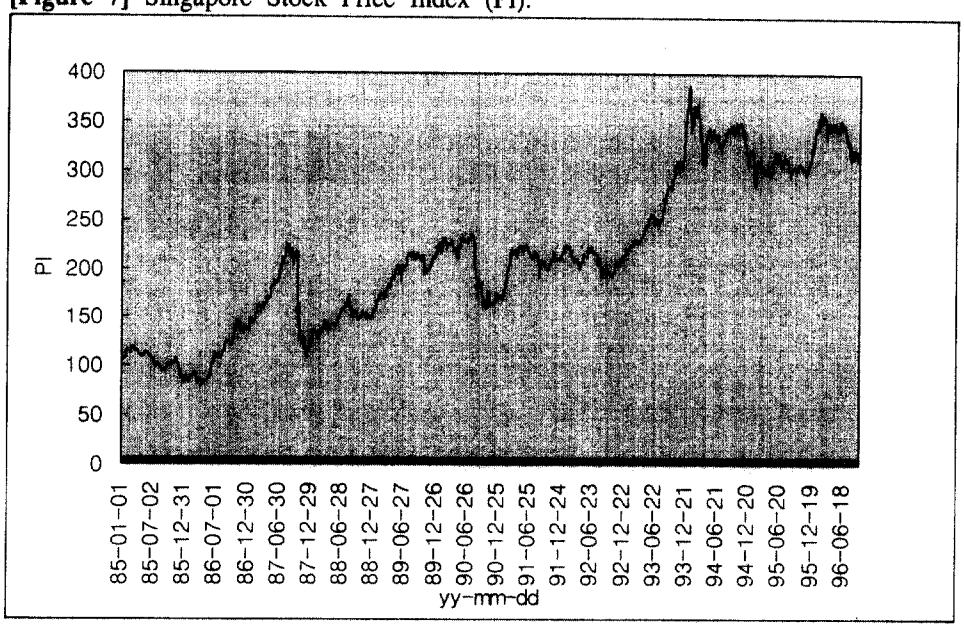
data points from Jan. 1, 1985 to Dec. 31, 1995, while the testing phase consisted of 186 data points from Jan. 1, 1996 to Sep. 16, 1996. The 5 input variables were transformed by a logarithmic transformation (L), a differencing operation (D), and a standardization operation (Z) as appropriate.

[Table 1] Description of the original variables. The training data were daily observations from Jan. 1, 1985 to Dec. 31, 1995; the test data from Jan. 1, 1996 to Sep. 16, 1996.

Label	Name	Description
PI	Stock Price Index	Singapore stock price index (1980 = 100).
RI	Total Return Index	Cumulative stock index return, incl. dividends.
DY	Dividend Yield	Dividend yield of Singapore stock market.
VO	Turnover by Volume	Trading volume of Singapore stock market.
PE	Price/Earnings ratio	Price/Earnings ratio for the Singapore stock index.

Model construction. Figure 7 shows a temporal plot of the Singapore stock price index. The changes in the stock price index can be categorized into quantized classes. In particular, the current study employed an APN comprising 6 elementary PNNs. Consequently, there were 7 categories which were labeled Levels 0 through 6. The class boundaries are listed in Table 2.

[Figure 7] Singapore Stock Price Index (PI).

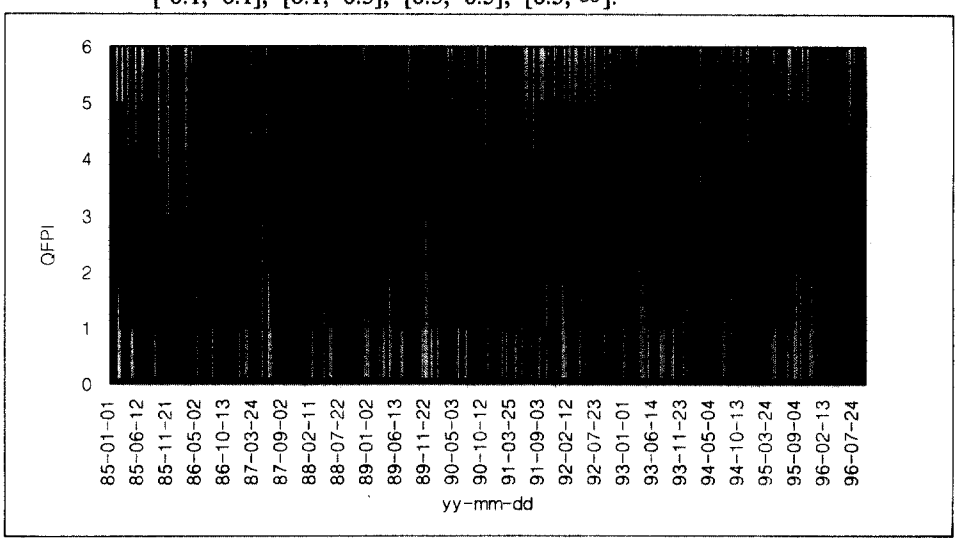


[Table 2] Quantized Levels of the percentage change from one period to the next. The arrayed probabilistic network (APN) contains elementary modules for each of Levels 1 through 6. If all forecasts are negative, then the default decision corresponds to Level 0.

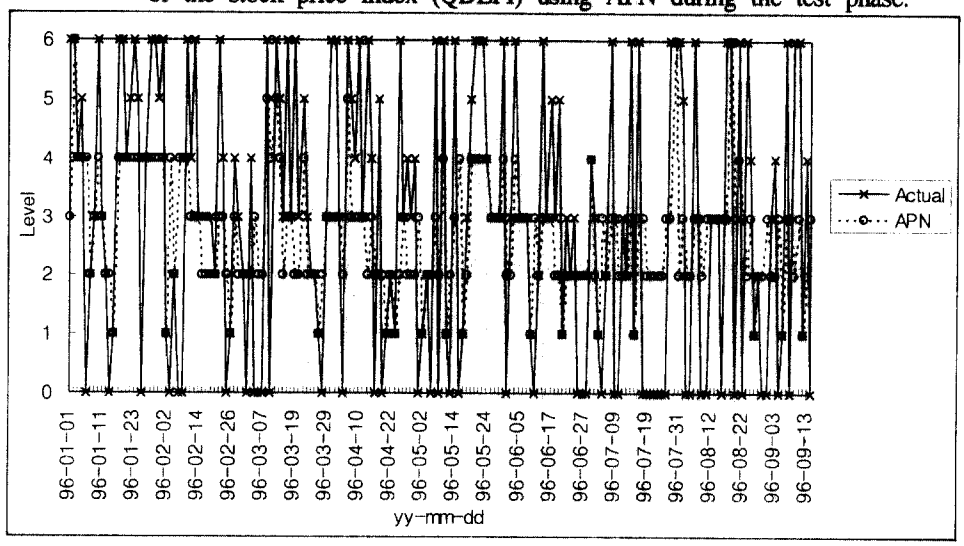
Level	Range (%)
0	(- ∞, -0.5%)
1	[-0.5%, -0.3%]
2	[-0.3%, -0.1%]
3	[-0.1%, 0.1%]
4	[0.1%, 0.3%]
5	[0.3%, 0.5%]
6	[0.5%, ∞]

Figure 8 depicts the discretized form (QFPI) of the FPI. Figure 9 plots the actual and predicted values of the quantal change in the differenced log of the price index (QDLPI) due to APN during the test phase. Figure 10 compares the actual QDLPI against forecasts due to the BPN(5*15*1) model: that is, a backpropagation model with 5 nodes in the input layer, 15 in the hidden layer, and 1 output node. The next chart, Figure 11, compares the actual QDLPI against forecasts from an RNN model. Figure 12 shows the actual and forecast values of QDLPI due to CBR.

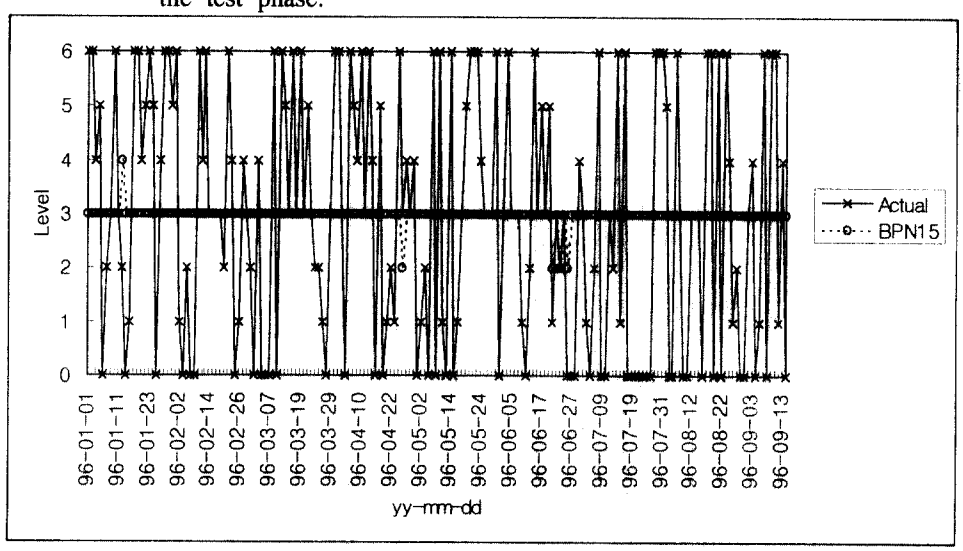
[Figure 8] Quantized fractional change (QFPI) in the Singapore Stock Price Index over the entire timespan covering both the training and testing phases. The categories or Levels are as follows : (- ∞, -0.3), [-0.3, -0.1], [-0.1, 0.1], [0.1, 0.3], [0.3, 0.5], [0.5, ∞].



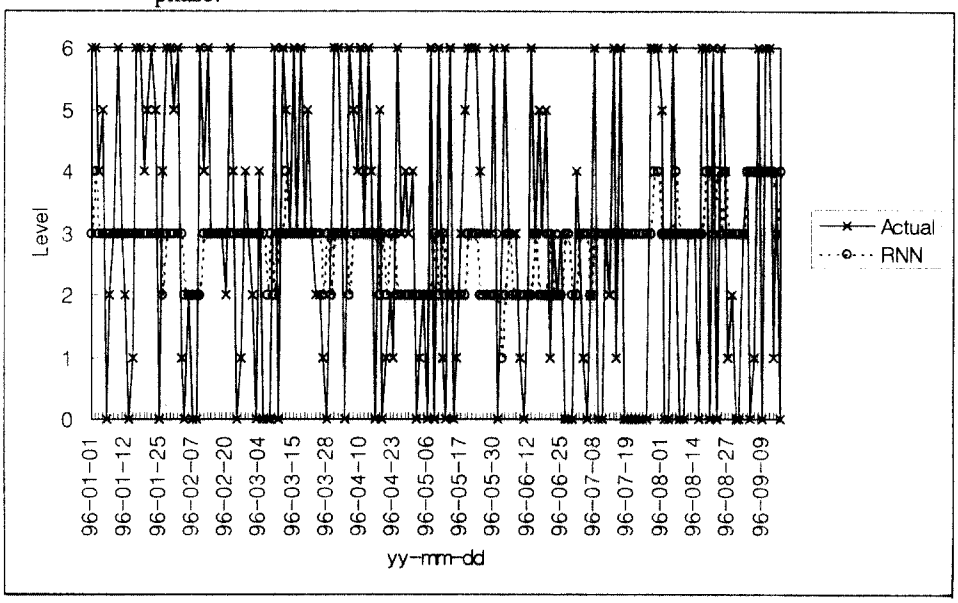
[Figure 9] Actual vs. predicted values of the quantal change of the differenced log of the stock price index (QDLPI) using APN during the test phase.



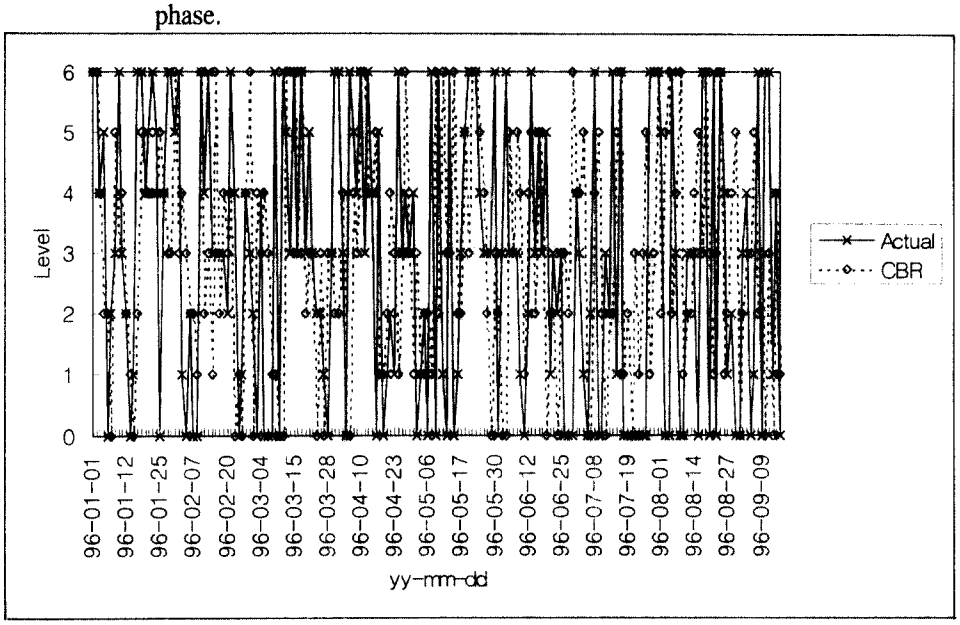
[Figure 10] Actual vs. predicted values of the quantal change of the differenced log of the stock price index (QDLPI) using BPN(5*15*1) during the test phase.



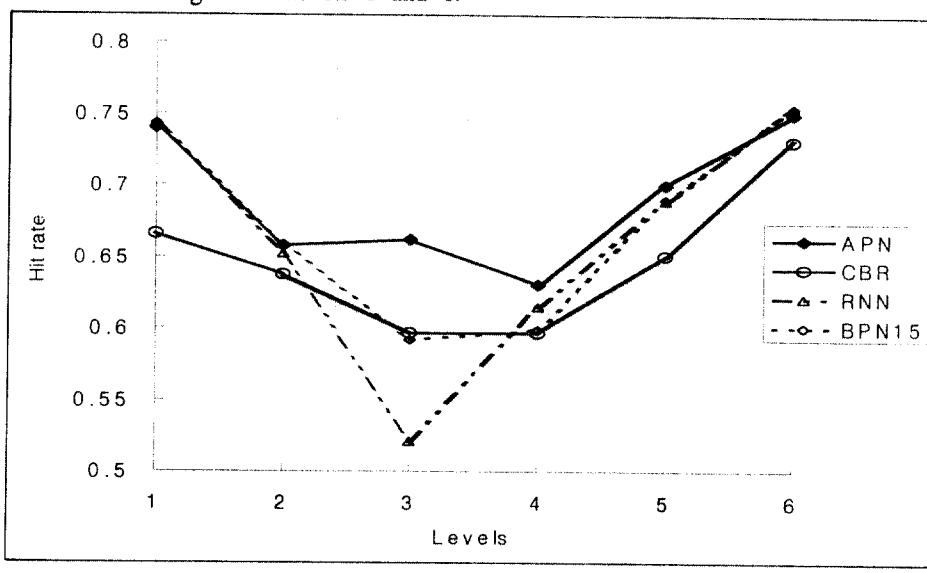
[Figure 11] Actual vs. predicted values of the quantal change of the differenced log of the stock price index (QDLPI) using RNN during the test phase.



[Figure 12] Actual vs. predicted values of the quantal change of the differenced log of the stock price index (QDLPI) using CBR during the test phase.



[Figure 13] Hit rate by Level for different learning architectures. Level 3 corresponds to a lack of significant movement in the market index; in this situation the fluctuations may be largely random with minor prices changes neglecting the "lumpiness" of purchases and sales in the market. On the other hand, Levels 1 and 6 pertain to larger movements which are more likely to be precipitated by significant news. Meanwhile, Levels 2 and 5 lie somewhere in the middle of the spectrum of activity: namely, the noise of Level 3 versus the signal of Levels 1 and 6.



Results of study. The performance among the predictive models is presented in Table 3. The metric of accuracy is the hit rate, or proportion of correct forecasts.

[Table 3] Hit rate by Level among the forecasting modules. The 3 BPN models yield comparable performance.

Model \ Level	1	2	3	4	5	6
APN	0.7433	0.6577	0.6612	0.6344	0.6989	0.7526
CBR	0.6666	0.6379	0.5967	0.5967	0.6505	0.7311
RNN	0.7433	0.6524	0.5215	0.6149	0.6898	0.7540
BPN(5*10*1)	0.7433	0.6577	0.5828	0.6042	0.6898	0.7540
BPN(5*15*1)	0.7433	0.6577	0.5913	0.5989	0.6898	0.7540

The patterns are easier to discern in Figure 13. It would appear that APN exhibits the best performance overall. This proposition is supported by Table 4, which indicates that APN is significantly better than RNN and superior to the best BPN(5*15*1) for Level 3 forecasts. On the other hand, Table 5 indicates that APN is mildly superior to BPN for Level 4, but not significantly so. In addition, the next table supports a similar conclusion concerning APN's superiority over CBR.

[Table 4] Tests of the difference in proportions among pairs of models for Level 3.

Models	Proportions	p-value	Decision
APN vs. RNN	0.6612 vs. 0.5215	0.0062	Reject Ho
APN vs. BPN(5*15*1)	0.6612 vs. 0.5913	0.1646	Reject Ho
CBR vs. RNN	0.5967 vs. 0.5215	0.1406	Accept Ho

[Table 5] Two-sample test for the difference in proportions for Level 4.

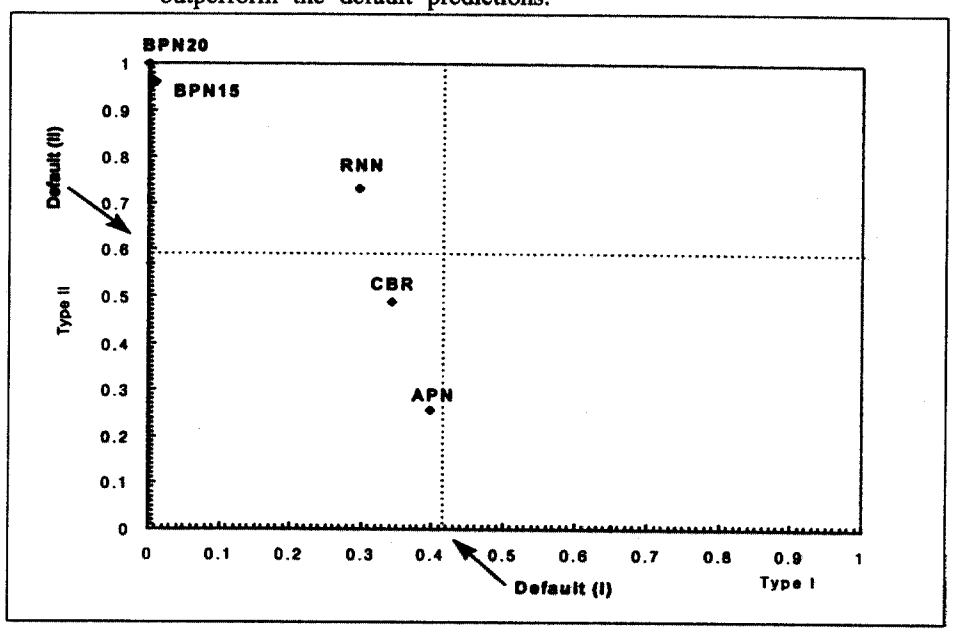
Models	Proportions	p-value	Decision
APN vs. BPN(5*15*1)	0.6344 vs. 0.6042	0.5486	Accept Ho

[Table 6] Two-sample test for the difference in proportions for Level 5.

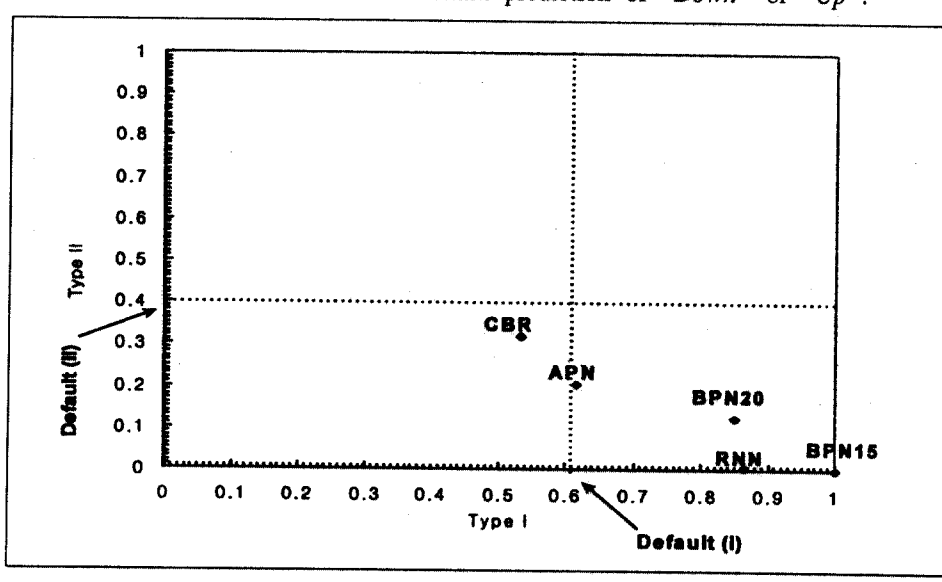
	Proportions	p-value	Decision
APN vs. CBR	0.6989 vs. 0.6505	0.3174	Accept Ho

Unfortunately the hit rate alone is an inadequate measure of performance since it ignores the extent of mistakes. The results of the Type I and Type II errors for each learning model at Level 3 are readily comprehensible in the form of Figure 14. More specifically, CBR and APN are the only models which supercede default performance in terms of minimizing both Type I and Type II errors. The results of the errors by model for a Level 4 change in the market indexes are charted on Figure 15, which reveals that CBR is the only model to outperform default prediction. Finally, the results of the errors by model for a Level 5 change in the market is given in Figure 16. The chart shows that CBR supercedes the other models, and in fact is the sole model with superior performance according to mistakes of both Type I and Type II.

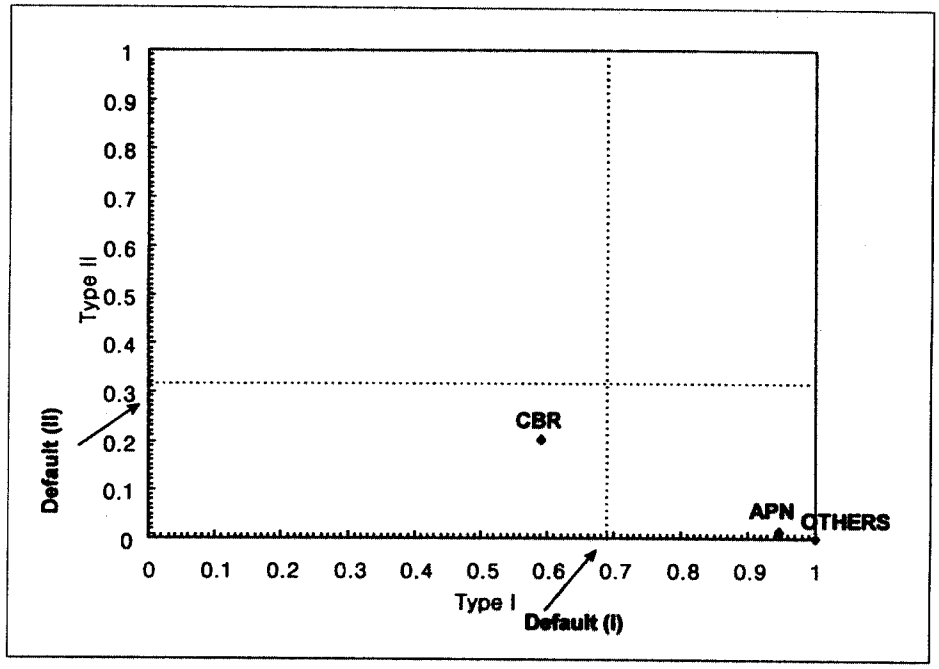
[Figure 14] Mistake chart for Level 3 predictions. Dashed lines indicates default mistakes based on a constant prediction of "*Down*" or "*Up*". For instance, Default (I) is the expected Type I error due to a constant forecast of "Down". Only models in the southwest corner outperform the default predictions.



[Figure 15] Mistake chart for Level 4 predictions. Dashed lines indicates default mistakes based on a constant prediction of "*Down*" or "*Up*".



[Figure 16] Mistake chart for Level 5 predictions. Dashed lines indicates default mistakes based on a constant prediction of "*Down*" or "*Up*".



In conclusion, the arrayed probabilistic network tends to outperform recurrent and backpropagation networks. However, case based reasoning tends to supercede the arrayed probabilistic network as well as the other techniques when mistakes are taken into consideration.

V. FUTURE WORK

A promising direction for the future is a comprehensive strategy using filtering and multistrategy learning. Time series data may be conditioned by filtering methods and then injected into learning techniques to predict financial variables.

Another direction for the future lies in the simultaneous prediction of multiple time series. In this way, more specific forecasts can be generated by constructing a multisector model which distinguishes among various primary, manufacturing, and service industries.

REFERENCES

Anderson, J. A. and Rosenfeld, E., eds. *Neurocomputing*. Cambridge, MA: MIT Press, 1988.

Grossberg, S. "Adaptive Pattern Classification and Universal Recoding: Part I. Parallel Development and Coding of Neural Feature Detectors." *Biological Cybernetics*, v. 23(3), 1976, pp. 121-134.

Hebb, D. O. *The Organization of Behavior*. NY: Wiley, 1949.

Hopfield, J. J. "Neural Networks and Physical Systems with Emergent Collective Computational Abilities." *Proc. Nat. Acad. Sciences USA*, v. 79(8), 1982, pp. 2554-2558.

Kohonen, T. *Self-Organization and Associative Memory*. NY: Springer-Verlag, 1984.

Rosenblatt, F. *Principles of Neurodynamics*. NY: Spartan, 1962.

Rumelhart, D. E., G. E. Hinton, R. J. Williams. "Learning Internal Representations by Back Propagation." In D. E. Rumelhart, J. L. McClelland and the PDP Research Group, *Parallel Distributed Processing*, v.1: Foundations, Cambridge, MA: MIT Press, 1986, pp. 318-362.

Sprecht, Donald F. "Probabilistic Neural Networks." *Neural Networks*, v. 3, 1990, pp. 109-118.