12-1-1992

Feature Subset Select ion and Financial Predict ion Tasks

W. Hsu
Purdue University School of Electrical Engineering

L. S. Hsu
Purdue University School of Electrical Engineering

M. F. Tenorio
Purdue University School of Electrical Engineering

FEATURE SUBSET SELECTION AND FINANCIAL PREDICTION TASKS

W. HSU
L. S. HSU
M. F. TENORIO

TR-EE *92-53*
DECEMBER 1992

SCHOOL OF ELECTRICAL ENGINEERING
PURDUE UNIVERSITY
WEST LAFAYETTE, INDIANA 47907-1285

Feature Subset Selection and Financial Prediction Tasks

W. Hsu L. S. Hsu M. F. Tenorio

Parallel Distributed Structures Laboratory
Department of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

Abstract

One major problem in applying neural networks to financial applications is the large number of features involved. The feature set is large because we simply do not know which of the given features may be useful to the system and include rather than risk throwing away a potentially useful one. In practice, training with the full set of features usually introduces unnecessary complexity and often degraded prediction performance. An important contribution of this paper is to focus the attention of neural network researchers on the need for a systematic feature preprocessing methodology for the purpose of improving predictability. The approach taken in this paper is to select subsets of the full feature sets that improve the prediction.

We discuss two different feature subset selection algorithms : the Penalty selection algorithm and the feature elimination algorithm. We explain the criterion *MisMatch* we use to evaluate a feature set. Both of the proposed algorithms are described using this criterion. Improved accuracies are obtained with both of the subsets compared to using all the features on the **DM-US** exchange rate Tuesday return prediction task. We describe current evaluation of the proposed techniques for actual trading.

Keywords: Feature Reduction, Feature Ranking, Financial Forecast.

1 Introduction

Traditionally, forecasting techniques of economic time series involves linear models[MW78, BJ70]. Recently, several researchers have suggested that financial systems may be better modeled by non-linear models because of the presence of chaos[BS88, Pet91]. Since then, many neural network algorithms have been applied to the financial application domain. Currently, most of the neural network approaches use some variants of the popular Backpropagation algorithm[RHW86] which is an iterative algorithm for training multilayer networks. However, these approaches suffer from the problem of long training time in light of the large amount of training data available in financial prediction applications. In practice, the difficulties with these approaches also include not knowing the "size" of the network and the stopping criteria for the algorithm.

Nevertheless, many successful attempts have been reported [WRH91, RABCK93, ea90]. Weigend and Refenes reported successes with the foreign currency exchange rate prediction problem using two variants of the Backpropagation algorithm[WRH91, RABCK93]. Their methods take as input the full set of training patterns obtained from the time series by a technique known as windowing[RABCK93]. A problem with using these unprocessed training data is that noise can result in slow training and degraded prediction performance.. An important contribution of this paper is to focus the attention of neural network researchers on the need for a systematic feature preprocessing methodology for the purpose of improving predictability. The approach taken in this paper is to select only subsets of the full feature sets that improve the prediction.

The organization of this paper is as follows. In Section 2, the feature subset selection problem is formulated formally. In Section 3, the two algorithms to feature subset extraction are described. In

Section 4, the German Mark (DM) versus the U.S. Dollar (US) exchange rate prediction problem is formulated. In Section 5, Experimental results of our approach are discussed and compared.

2 Problem Definition

Given a finite training set $S = (f, y)$ where $f \subseteq R^L$ and $y \subseteq R^1$, the learning algorithms associates the following

$$\mathbf{f}_1 \rightarrow y_1 \mathbf{f}_2 \rightarrow y_2 \vdots \mathbf{f}_n \rightarrow y_n$$

The vector f is called the full feature set. In the context of time series prediction, f consists of two parts, i.e.,

$$\mathbf{f} = [\mathbf{z}, \mathbf{i}] \tag{1}$$

where z is called the delayed vectors and consists of past samples of the time series itself. Typically, the delayed vectors are extracted from the time series by a technique called "windowing"[RABCK93]. i is a vector consisting of indicators or features potentially useful for the prediction problem. Each component of f is a feature. Each f is associated with a corresponding desired prediction y. The learning or training task is then to associate the f s with the y. We say a prediction is made when a novel f is presented to the network and a response is returned.

Following the definitions in [TR93], two vectors $[f^i, y^i]$ and $[\mathbf{f}^j, y^j]$ are *malicious* if

$$\|\mathbf{f}^i - \mathbf{f}^j\| \leq \alpha \quad \text{and} \tag{2}$$

$$\|y^i - y^j\| \geq \delta \tag{3}$$

where $a, 6 \in R^1$. Our methods seek to remove as much "maliciousness" from the training set as possible. In the next subsection, we describe various existing approaches to remove "maliciousness".

2.1 Related Work

Several neural network researchers have looked at this problem of improving prediction by removing "maliciousness" using various feature selection methods. Wong[WT91] uses the genetic algorithm to search the space of subsets of the full set of features. The fitness criterion used for evaluation is the actual prediction error made. This is an expensive operation since the learning architecture uses some variant of the Backpropagation algorithm.

A different approach to this problem is to remove some training samples (as opposed to features) from the training set. The candidate training samples to be removed belong to the "malicious" category[TR93]. This method is complementary to the methods proposed in this paper.

Another approach is to extract features by making linear combinations of the features in the full feature set. We proposed a decision boundary method for doing this in [HHT92]. Our method favors features that discriminate between clusters rather than fidelity of representation like principal component analysis does[Wat65]. This techniques works well if none of the features in the full feature set are confusing to the predictor.

Several related techniques exist in the pattern recognition field[DH73, Fuk90, Mei72] dealing with feature subset selection. All of the existing methods handle the case when the features in the full feature set are all useful to the classification task. The motivation there is to select features until the further increase in prediction is not justified by the added complexity of having an extra feature. However in our application, the problem is quite different. We are typically given a large set of features constituting the full feature set where not all of the features helps and some features may confuse the predictor.

Another related technique is the well known analysis of variance (ANOVA)[Sam89]. This technique computes the sum of squares between classes SS(between) and the sum of squares within each class SS(within). This technique is described for situation where each class is characterized by only a single quantity or feature. The ratio of SS(between) with that of SS(within) can be used as an alternative criterion for both of the proposed feature selection algorithms. Our approach in

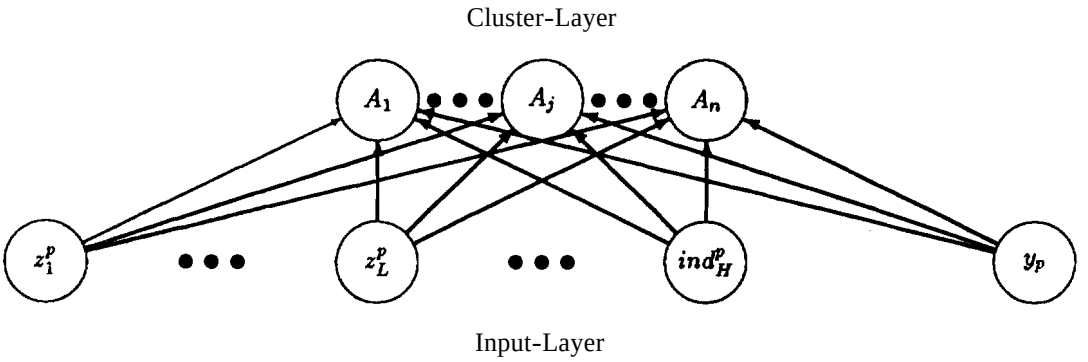

Figure 1: The SupNet Architecture

this chapter uses the number of *MisMatch* as our criterion. The advantage is that our criterion can handle cases when each class is characterized by more than one feature as is typically in many prediction tasks.

3 The Supervised Clustering Network

The input of the network is called the full feature vector denoted by f

3.1 Network Architecture

The supervised clustering network SupNet is a neural network that classifies a given set of n delayed vectors $\mathbf{z}^p$ into N clusters, according to their corresponding values of y_p. Its architecture is shown in Figure 1.

The network consists of 2 layers. The bottom layer is the input layer. It consists of L nodes each representing a feature. The last node represents the desired prediction **y.** We assume in this paper that y is a scalar.

The top layer is the cluster layer. Its size is determined dynamically by the learning algorithm described in the next subsection. With this architecture, each cluster node conceptually represents input vectors with similar values of **y.**

The weights connecting a given cluster node c to the input nodes are the components of the weight vector W^c. The values of these weight vectors are determined by the learning algorithm which will be described in the next subsection.

3.2 The Learning Algorithm

The learning is done in two stages. During the first stage, Each input vector f is masked to appear to be a zero vector and only the value of its corresponding y is presented. We follow the algorithm used in **ClusNet [Hsu92]** to determine the (L+1)-**th** component of the weight vectors for all the clusters. The first L components of W remain at zero. After this stage, the input vectors are grouped into clusters with respect to the desired prediction **y.**

During the second stage, the input vectors are presented a second time. Let us assume that when the *p*-**th** input vector $[\mathbf{f}^p, y^p]$ is presented, the c-th cluster node has the lowest activation

among existing cluster nodes. The activation of node c with respect to this input is

$$A_c = (y^p - W^c_{L+1})^2. \quad (4)$$

where W^c_{L+1} is the $L+1$-th component of the weight vector. We say that the c-th node is the winning node and the first L components of its weight vector is updated to:

$$W^c_i = \frac{1}{n_c}\{(n_c - 1)(W^c_i + z^p_i)\}, \quad 1 \leq i \leq L \quad (5)$$

where n_c is the number of input vectors belonging to cluster c including the input vector $[\mathbf{f}^p, y^p]$. This procedure is repeated for all the input vectors. At the end of this stage, the weight vectors W of the network are known. Each cluster is represented by its weight which is the mean of all input vectors that belongs to it.

Using the described algorithm above, each full feature vectors $\mathbf{f}^p$ are clustered with respect to their corresponding values of y_p. Consider two of these vectors, $\mathbf{f}^i$ and $\mathbf{f}^j$ belonging to a single cluster. The distance between them may be large even though they are in the same cluster because the clustering was done with respect to y. The basis of the two proposed feature subset selection algorithms is to eliminate features that have different values within a single cluster. The motivation is that the resulting feature subsets will enable better prediction.

For our experiments, as a measure of "maliciousness", we use a criterion called *MisMatch* which will be described next.

4 The Feature Selection Methodology

We describe how the proposed feature subset selection algorithms can be used in a specific prediction task.

1. Choose a set of Full Feature Vectors f. A portion of these are used as the training set, the rest are used as a prediction set.
2. Choose a small positive number ϵ. Using ϵ, the Full Feature Vector f in the training set are separated into three categories:
   (a) Category A consists of vectors whose corresponding $y > \epsilon$
   (b) Category B consists of vectors whose corresponding $y < -\epsilon$
   (c) Category C consists of all other vectors.

   The motivation for this particular setup will become clear in the application section.
3. Using SupNet, the sets of training vectors in Categories A and B are separately clustered according to the y value. The corresponding weight vectors for each cluster are also calculated.
4. After training, if a training vector that belongs to category A is predicted to be in Categories B or C, or a vector that belongs to category B is predicted to be in Categories A or C, then we say that a *MisMatch* has occurred.
5. The Penalty Feature Selection and Elimination Algorithms which will be described in the next subsections are applied to f to obtain feature subsets p and e respectively.
6. p or e are then used as input to a prediction algorithm.

4.1 The Penalty Feature Subset Selection Algorithm

The Penalty Feature Subset Selection Algorithm starts with the Full *feature* Vector f containing L components/features. Each feature is associated with a penalty which is computed as the number of misprediction due to the single feature. Features with large penalties are removed. from f. The resulting feature subset is called the Penalty Feature Vector p.

4.2 Single Cluster per Category

In the case when there is only one cluster in a category, the Penalty is calculated as follows:

Step 1 : Initialize **Penalty[1..k]** to zero.
Step 2 : Cluster the L components off
into N clusters and compute the weight vectors W^c for
$c = 1 .. N$.
Step 3 : Select component k=l
Select cluster c =1
Select vector i = 1
Step 4 : if vector $i \in c$ but $d \neq c$
/* vector i is misclassified to cluster d */, i.e.
$(f_L^{c,i} - W_L^c)^2 > (f_L^{c,i} - W_L^d)^2$ then
add 1 to **Penalty[k]**
end
Step 5 : repeat Step 4 for $i = 1 : N_c$
Step 6 : repeat Step 4 for c = 1 : N
Step 7 : repeat Step 4 for k = 1 to L

4.3 Multiple Clusters per Category

In the case when there are multiple clusters in a category, the Penalty is calculated as follows:

Step 1 : Initialize **Penalty[1..k]** to zero.
Step 2 : Cluster the L components off
into N clusters and compute the weight vectors W^c for
$c = 1 .. N$.
Step 3 : Select component k=l
Select cluster c =1
Select vector i = 1
Step 4 : if vector $i \in c$ but d and c belong to different Category
/* vector i is misclassified to different Category */, i.e.
$(f_L^{c,i} - W_L^c)^2 > (f_L^{c,i} - W_L^d)^2$ then
add 1 to **Penalty[k]**
end
Step 5 : repeat Step 4 for vectors i belonging to cluster c
Step 6 : repeat Step 4 for clusters c
Step 7 : repeat Step 4 for features k

4.4 The Feature Elimination Algorithm

The Feature Elimination Algorithm starts with the ***Full Feature Vector*** f containing ***L*** features. The number of ***MisMatch*** due to f is computed and stored. Subsequently, L − 1 feature subsets each containing L − 1 features are computed from f by removing one different feature at a time from f. These feature subsets are then evaluated to obtain the number of ***MisMatch*** due to each one of them. The subset resulting in the minimum number of ***MisMatch*** as well ***as*** the number of ***MisMatch*** are recorded. The process is then repeated until feature subsets containing only one feature is generated. When the process is terminated, the resulting feature subset, called the eliminated feature set denoted by **e**, is the recorded feature subset with the minimum number of ***MisMatch*** among all the recorded feature subsets.

The algorithm is explained in greater detail in the following pseudo code.

FUNCTION **FindMisMatch(f)**;
Step 1 : iff is of unit length, the base case of the recursion

is reached. Return.
Step 2 : Evaluate the number of mismatches due to f as follows
CurrentMisMatch = ComputeMisMatch(f).
Step 3 : Generate **length(f)-1** subfeatures denoted by $\mathbf{f}_i'$
where $\mathbf{f}_i'$ is obtained from f by removing feature i
Step 4 : Compute **MisMatch[i] = ComputeMisMatch($\mathbf{f}_i'$)** for
i = 1 : $length(\mathbf{f}) - 1$.
Step 5: Let the index of the minimum of **MisMatch[i]** be i_{min}.
Record the tuple (i_{min}, **MisMatch[i]**) in LIST.
Step 6 : Call **FindMisMatch** with $\mathbf{f}'_{i_{min}}$.
Step 7 : Let the index of the minimum of the MisMatch in LIST be $LIST_{min}$.
The Eliminated Feature Vector e can be reconstructed from
the **associated** i_{min} in LIST.

The above algorithm returns the ***Eliminated Feature Vector*** e found **by** the Feature Elimination Algorithm. The algorithm is independent of the number of clusters. However, it makes use of the function **ComputeMisMatch(f)** which does depend on the latter.

4.5 Single Cluster per Category

In the case when there is only one cluster in a category, the function **ComputeMisMatch(f)** is as follows:

FUNCTION **MisMatch(f)**
Step 1 : Initialize MisMatch = 0;
Step 2 : Cluster the L components of f
into N clusters and compute the weight vectors W^c for
c = 1 .. N.
Step 2 : Select cluster c =1
Select vector i = 1
Step 3 : if $i \in$ c but d $\neq$ c
/* vector i is misclassified to cluster d */, i.e.
$(\mathbf{f}_L^{c,i} - \mathbf{W}_L^c)^2 > (\mathbf{f}_L^{c,i} - \mathbf{W}_L^d)^2$ then
add 1 to MisMatch
end
Step 4 : repeat Step 4 for all vectors $i \in$ cluster c
Step 5 : repeat Step 4 for all clusters c
Step 6 : return MisMatch

4.6 Multiple Clusters per Category

In the case when there are multiple clusters in a category, the function **ComputeMisMatch** is changed to read:

FUNCTION **ComputeMisMatch(f)**
Step 1 : Initialize MisMatch = 0;
Step 2 : Cluster the L components of f
into N clusters and compute the weight vectors W^c for
c = 1 .. N.
Step 2 : Select cluster c =1
Select vector i = 1
Step 3 : if $i \in$ c but d $\neq$ c
/* vector i is misclassified to a different Category */, i.e.
(1) c and d belong to different Category and

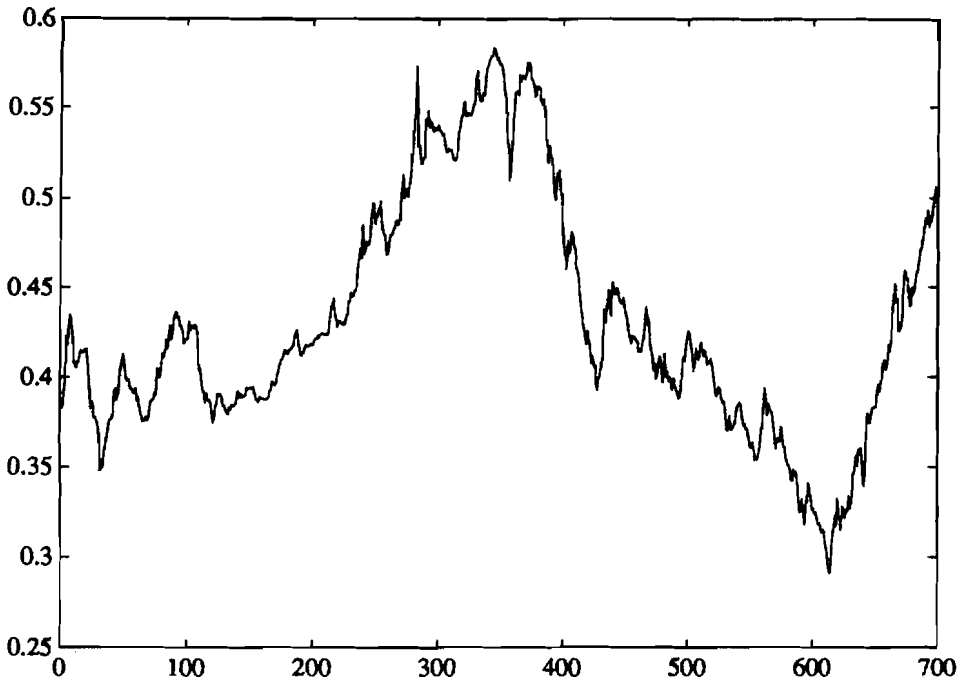

Figure 2: The DM-US exchange rate on Tuesdays for period 6/14/77 to 5/20/87

(2) $(\mathbf{f}_L^{c,i} - \mathbf{W}_L^c)^2 > (\mathbf{f}_L^{c,i} - \mathbf{W}_L^d)^2$ then
add 1 to MisMatch
end

Step 4 : repeat Step 4 for all vectors $i \in$ cluster c
Step 5 : repeat Step 4 for all clusters c
Step 6 : return MisMatch

5 The US-DM Exchange Rate Prediction Task

The primary purpose of this example is to demonstrate improved prediction using one of the feature subsets compared to that obtained with the the full feature vector f . Our experimental setup follows closely that used by Weigend[WRH91].

5.1 The Data

The foreign currency exchange data are taken from the Monetary Yearbook of the Chicago Mercantile Exchange. They are daily closing bids for five currencies (German Mark (DM), Japanese Yen, Swiss Franc, Pound Sterling and the Canadian Dollars) with respect to the U.S. Dollar. The German Mark (DM) with respect to the U.S. Dollar (US) is seen in Figure 2 for the period starting from September 1973 to May 1987.

5.2 The Next Day Returns

Instead of predicting the actual value of the DM US time series itself, our forecasting experiment predicts a quantity of particular interest to a currency trader called the returns of the next day, denoted by r_{t+1}. In order to make profitable moves in the foreign currency markets, the trader must predict the sign of the next day returns which we define as

$$r_{t+1} = p_{t+1} - ave(p_{t-k}, \ldots, p_t) \tag{6}$$

where k represents a number of days which are averaged. This quantity r_{t+1} represents the movement of the price p_{t+1} in relation to the average of the past k days, i.e.,

$$sign(r_{t+1}) = \begin{cases} 1 & \text{if } r_{t+1} > \epsilon \\ -1 & \text{if } r_{t+1} < -\epsilon \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

We preserve the "day of the week" effect by predicting only the returns on Tuesdays in order to avoid averaging the dynamics of different days of the week[Hsi89]. The data set is split into two sets as shown in Table 1.

name	from	to	size
train	Sept 1973	April 1983	500
test	April 1983	May 1987	212

Table 1: Data Sets for Currency Exchange Rate Predictions

The Tuesday returns for the training set is plotted in Figure 3. The same quantity for the testing set is plotted in Figure 4.

We group the next day return into the following categories as shown in Table 2. Category C

Category	sign(r_{t+1})	Remarks
A	positive	Uptrend
B	negative	Downtrend
C	don't cares	-

Table 2: Classification of the time series patterns into three categories

patterns refers to those days when the prices hardly moves from one day to another. There is not much to be gained or lost by trading on those days. Category A patterns represent days in which the prices are going up significantly. This class of days are of interest to the trader because there is profit to be made. Category B patterns represent days when there is significant downward movement of the market price. This class of days are equally important because traders with this information can short positions in the market. The decision between Category A and Category C patterns are made by the parameter ϵ.

5.3 Potentially Useful Features

The definitions of the two quantities that are commonly used for economic predictions are as follows.

- The k day *trend* at day t is the running mean of the returns of the k last days

$$trend_{k,t} = \frac{1}{k} \sum_{t-k+1}^{t} r_t = p_t - p_{t-k+1} \tag{8}$$

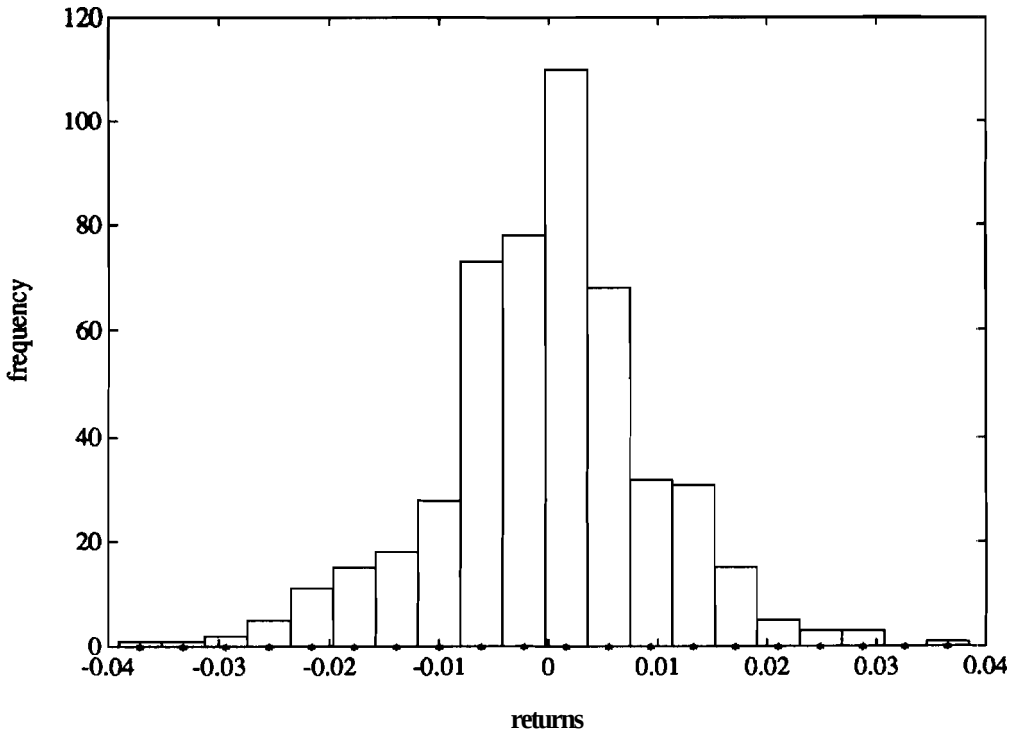

Figure 3: The DM-US exchange rate returns on Tuesdays for the training set

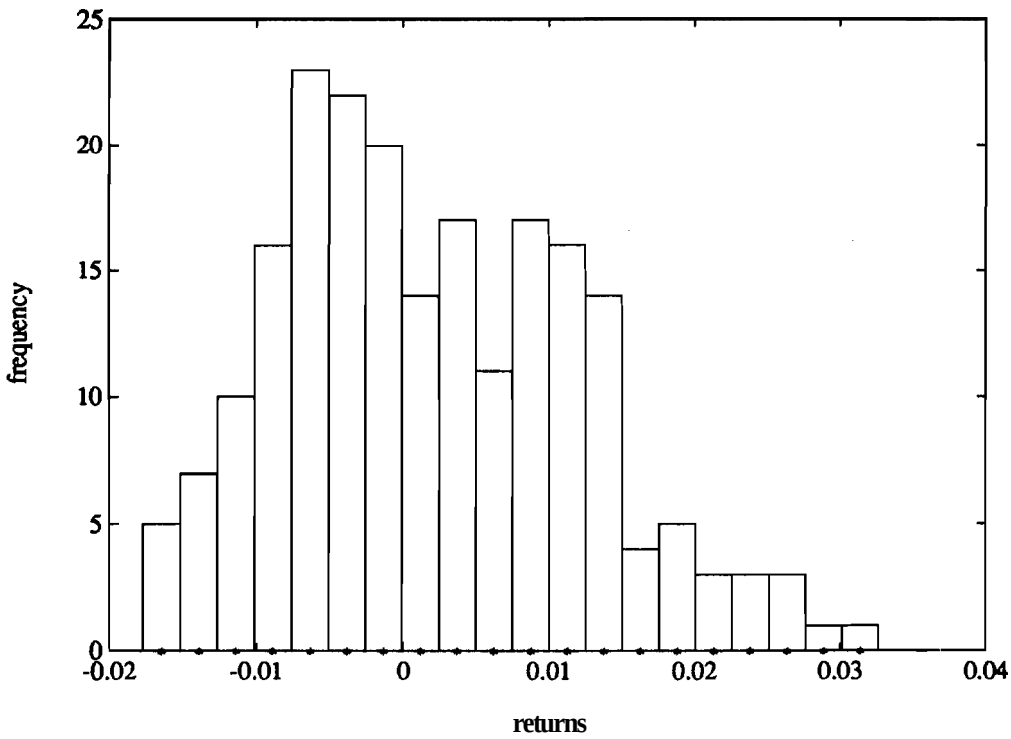

Figure 4: The DM-US exchange rate returns on Tuesdays for the test set

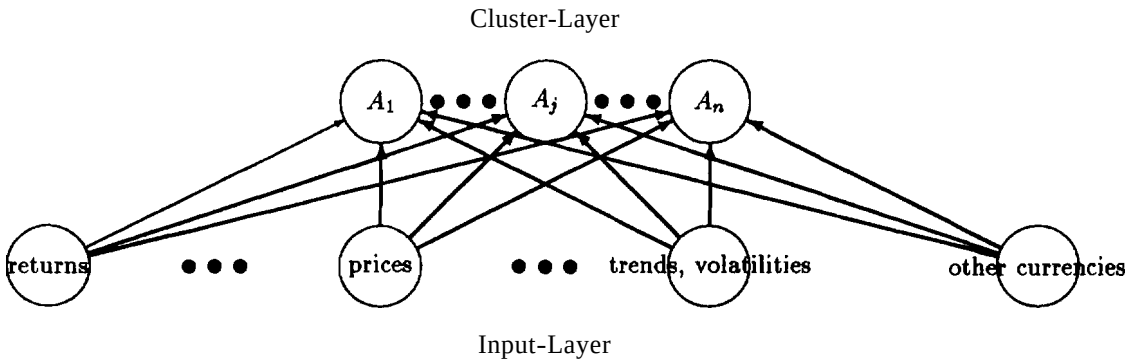

Figure 5: The SupNet Architecture

- Similarly, the k ***day volatility*** is the running standard deviation of the returns of the k last days,

$$\sqrt{\frac{1}{k-1}\sum_{t-k+1}^{t}(r_t - trend_{k,t})^2} \tag{9}$$

The k-day tangent slope indicator is computed from a time series by fitting a straight line $y = mx + c$ from the k-th past day to the current day. The tangent slope indicator is the value of m obtained. The ratio of two k-day tangent slope indicators with different k captures the turning characteristics of the time series.

The **SupNet** architecture is given in Figure 5. It contains inputs for past week of daily DM returns, value of the five major currencies for Monday and the returns of the 5 major currencies. Both the volatilities and trends are computed for various window sizes between one week (k=5) and three months (k=65). The prices themselves are also input in case they too play a role in the dynamics of the returns. Trends for the other currencies are also input. Three tangent slope indicator on the DM price is also input. In typical financial applications, the number of features given for a prediction task may be even larger because we simply do not know which features may help the system and include rather than risk throwing away a potentially useful feature.

Some of these features represent completely different quantities measured using unrelated scales. To prevent one feature from overwhelming the contribution of other features, we normalize feature j by the following quantity n_j computed as

$$n_j = \sum_c \sum_i (\mathbf{X}_j^{c,i} - \mathbf{W}_j^c)^2$$

where c represents clusters and i represents the vectors belonging to cluster c.

5.4 The SupNet Prediction Procedure

The prediction procedure is described as follows :

1. The feature subset is presented to **SupNet** and we compute D_A and D_B as the activation of the winner node in Category A and Category B respectively.
2. **If** $||D_A - D_B|| \leq \epsilon_m$, we predict X as belonging to Category C.

3. Otherwise, if $D_A > D_B$, we output a **prediction** of Category B.

4. Otherwise, if $D_B > D_A$, we output a prediction of Category A.

The parameter ϵ_m is related to the choice of ϵ. For all our experiments, we compute ϵ_m as follows.

$$\epsilon_m = \frac{1}{N_{\text{Category C}}} \sum_c \sum_{i=1}^{N_c} (\mathbf{X}^{i,c} - \mathbf{W}^c)^2 \delta(\mathbf{X}^{c,i}, C) \tag{10}$$

where

$$\delta(\mathbf{X}^{c,i}, C) = \begin{cases} 1 & \text{if } \mathbf{X}^{c,i} \in \text{Category C} \\ 0 & \text{otherwise} \end{cases}$$

and $N_{\text{Category C}}$ is the number of $\mathbf{X}^{c,i} \in$ Category C. If the value of ϵ_m is too large, patterns belonging to Category A and B will be drawn into Category C. If the value is too small, then patterns in Category C will be predicted as belonging to category A or B. In our experiments, a procedure is used to dynamically seek a suitable value for ϵ_m.

6 Empirical Results

6.1 Approach 1 : Single Cluster Per Category

For a specific ϵ, the patterns in the data is broken down into categories as shown in Table 3.

Category	Training Set	Testing Set
A	134	78
B	136	61
C	230	73

Table 3: Grouping the Returns Data into Categories

Feature Set	length	Category A	Category B	Category C
The ***Full Feature Vector*** f	71	47	28	35
The ***Penalty Feature Vector*** p	32	50	27	48
The ***Eliminated Feature Vector*** e	11	65	32	69

Table 4: Prediction Performance for the Single Cluster Per Category Approach

From Table 4, using all the features in the ***Full Feature Vector,*** about chance prediction is recorded. Using the the penalty feature subset p, and eliminated feature subset e, improved prediction is observed.

In this experiment, the best prediction percentage for each Category is 83.3, 52.4, 94.5 respectively. Another measure of error that is appropriate for this data is the ***critical misprediction error count*** defined as the number vectors that is associated with **uptrends** but are predicted to be associated with downtrends and vice versa. In this experiment, the number of ***critical misprediction error count*** is 7 for f, 2 for p and 0 for e.

6.2 Approach 2 : Multiple Cluster per Category

In this method, we allocate multiple clusters per Category. The results are as shown in Table 5. In Table 5, we observe the best prediction percentage for each Category is 82.1, 67.2 and 84.9

Feature Set	length	Category A	Category B	Category C
The ***Full Feature Vector*** f	71	37	13	46
The ***Penalty Feature Vecior*** p	32	52	34	33
The ***Eliminated Feature Vector e***	14	64	41	62

Table 5: Prediction Performace for the Multiple Clusters Per Category Approach

respectively. In this experiment, the ***critical misprediction error count*** is 10 for f, 9 for p and 0 for e.

The Eliminated Feature Vector e obtained in the multiple cluster experiments contains the following features that are important for this predicting task including

- the 2 and 5 point tangent slope indicator
- the past one week returns of the DM,
- the trends for 5, 10 and 65 days on the DM,
- 65 day trend on the swiss franc,
- the 20 and 40 day volatility
- the Monday returns on the British Pound
- the Monday returns on the Japanese Yen.

Upon closer analysis, we notice that features associated with the Canadian Dollar are often being eliminated by both of our subset selection algorithms. This finding is consistent with the observations of other researchers[WRH91, LeB91]. This could be due to the strong coupling between the Canadian and U.S. economies.

In this paper, we described and demonstrated the use of the Feature Selection and Elimination Algorithm for Prediction. Several factors favor our prediction methodology.

- Our assumption that patterns associated with small value of y for this particular prediction tasks appear to help remove some malicious examples from the training set.
- The proposed Penalty Feature Subset Selection and Elimination Algorithm are effective in removing features and resulting in improved prediction ability.
- The built-in parameter ϵ_m allows us to fine tune the tradeoff the accuracy of the predictions in the various Categories as required by different prediction tasks..

7 Conclusions and Further Work

Current neural network approaches to financial applications typically uses some variants of the Backpropagation algorithms. In practice, these methods have several difficulties including long training time due to the often large training set and the stopping condition is not known for each application. Perhaps the main obstacle to the use of neural networks techniques in actual financial applications is its "black-box" nature which makes it difficult to understand why a certain decision or prediction is made. Our proposed network called **ClusNet**[Hsu92] represents an alternative training method for financial applications.

In this paper, we use the **SupNet** architecture to learn to predict the sign of the returns of the US-DM exchange rate. In our experiments, both feature subsets obtained with our algorithms result in more accurate prediction performance on the exchange rate prediction task than the unprocessed full feature set. The Feature Elimination algorithm requires more execution time but its resulting

feature subset enables more accurate prediction performance. Further research is on-going to develop and compare other feature subset algorithms. An important contribution of this work is to focus the attention of neural network researchers on the need for a systematic feature preprocessing methodology for the purpose of improving predictability.

References

[BJ70] G. E. P. Box and G. M. Jenkins. *Time series analysis, forecasting and control.* Holden-Day, 1970.

[BS88] W. A. Brock and C. L. Sayers. Is the business cycle characterized by deterministic chaos? *Journal of Monetary Economics,* 22:71–90, 1988.

[DH73] R. O. Duda and P. E. Hart. *Pattern classification and scene analysis.* John Wiley and sons, 1973.

[ea90] T. Kimoto et al. Stock market prediction with modular neural networks. In *IJCNN* 90 *(San Deigo)*, 1990.

[Fuk90] K. F'ukunaga. *Statistical Pattern Recognition.* Academic Press Inc., 1990.

[HHT92] W. Hsu, L. S. Hsu, and M. F. Tenorio. A decision boundary method for embedding selection. Technical Report TR-EE-92-xx, Purdue University, 1992.

[Hsi89] David Hsieh. Tetsing for nonlinear dependence in daily foreign exchange rates. *Journal of Business,* pages 339–369, 1989.

[Hsu92] William Hsu. *Nonlinear and self adapting methods for prediction.* PhD thesis, Purdue University, 1992.

[LeB91] B. LeBaron. Technical trading rules and simulated processes for foreign exhcnage rates. Technical report, Department of Economics, University of Wisconsin, Madison, 1991.

[Mei72] W. Meisel. *Computer-Oriented Approached to Pattern Recognition.* Academic Press, 1972.

[MW78] S. Makridakis and S. C. Wheelwright. *Forecasting : methods and applications.* John Wiley and Sons, 1978.

[Pet91] E. E. Peters. *Chaos and Order in the capital markets.* Wiley, 1991.

[RABCK93] A. N. Refenes, M. Azema-Barac, L. Chen, and S. A. Karoussos. Currency exchange rate prediction and neural network design strategies. *Journal of Neurocomputing and Applications,* 1(1), 1993.

[RHW86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representation by error propagation. In *Parallel Distributed Processing : Explorations in ihe microstructure of cognition,* volume 1. MIT Press, 1986.

[Sam89] M. L. Samuels. *Statistics for the life sciences.* Dellen Publishing Company, 1989.

[TR93] E. Tuv and A. N. Refenes. Removal of catastrophic noise in hetero-associative training samples. In *Proc. IJCNN 93 (Japan).* (to submit), 1993.

[Wat65] S. Watanabe. Karhunen-loeve expansion and factor analysis – theoretical results and applications. *4th Proc. Conf. Info. Theory,* 1965.

[WRH91] A. S. Weigend, D. E. Rumelhart, and B. A. Huberman. Predicting sunspots and exhange rates with connectionist networks. In M. Casdagli and S. Eubank, editors, *Nonlinear modeling and forecasting,* pages 395–432. Addison-Welsey, 1991.

[WT91] **F. Wong and P. Y. Tan. Neural networks and genetic algorithm for economic forecasting. AI *in Economics and Business Administration,* 1991.**