2008

# A Comparitive Study of Supervised and Unsupervised Learning Methods in Forecasting the U.S. 30-Year Treasury Bond Yield

Nicole Andrea Powell

FLORIDA STATE UNIVERSITY

FAMU-FSU COLLEGE OF ENGINEERING

# A COMPARITIVE STUDY OF SUPERVISED AND UNSUPERVISED LEARNING METHODS IN FORECASTING THE U.S. 30-YEAR TREASURY BOND YIELD

By

NICOLE ANDREA POWELL

A Thesis submitted to the
Department of Electrical and Computer Engineering
in partial fulfillment of the
requirements for the degree of
Master of Science

Degree Awarded:
Fall Semester, 2008

The members of the Committee approve the Thesis of Nicole Andrea Powell defended on October 24, 2008.

Simon Y. Foo
Professor Directing Thesis

Anke Meyer-Baese
Committee Member

Mark H. Weatherspoon
Committee Member

Approved:

Victor DeBrunner, Chair, Electrical and Computer Engineering

C.J. Chen, Dean, FAMU-FSU College of Engineering

The Office of Graduate Studies has verified and approved the above named committee members.

For my parents, James and Angela, as well as my entire family. Thank you for the love and support.

## ACKNOWLEDGEMENTS

I would like to thank my thesis advisor Dr. Simon Foo for allowing me the opportunity to formulate a great topic. I would also like to thank him for his constant guidance and support throughout the course of both my undergraduate and graduate degrees.

I would like to thank Dr. Anke Meyer-Baese and Dr. Mark Weatherspoon for their help and guidance. I am grateful to them both for the support they have provided over the course of my thesis writing.

Also, I would like to thank Dr. Gary Benesh and Dr. Pamela Coats of Florida State University’s College of Finance for their help in obtaining data and narrowing the topic. Last but not least, thank you to all the professors who have contributed tremendously to my academic success.

## TABLE OF CONTENTS

## LIST OF TABLES

# LIST OF FIGURES

# ABSTRACT

The prediction of any aspect of the future has always fascinated mankind because of the possible benefits of this knowledge, especially financial benefits. From year to year, many stockholders would like to be able to know if the price of their commodity will increase or decrease, and in turn this prediction may help them in the decision to buy more or sell what they currently have. It is widely known that the stock market is a volatile and complex entity which is affected by various factors such as government policies, political situations, public events, internal company politics and much more. However there is no way of knowing exactly which factor will affect stock prices and how much the price will be affected.

Financial trend forecasting is a major component in corporate finance because predictions of future prices, indices, volumes and several other values are often incorporated into the economic decision-making process for a particular company. For the average investor financial trend forecasting would mean a greater profit (or smaller loss). To recognize specific trends for forecasting capabilities, it is important to develop a method for eliminating speculation and to investigate new algorithms for detecting patterns.

Although there are many different approaches available, in this thesis a comparison between an unsupervised classification technique, namely K-means clustering, and supervised learning algorithms, namely support vector machines and radial basis functions, will be performed. The three pattern recognition systems will be tested against real-world data concerning the U.S. 30-Year Treasury bond yield. Determining the yield trend is approached as a technical analysis problem for this particular study: ignoring underlying factors and focusing on finding patterns directly from historical data.

The results from each of the three networks are compared and analyzed. The performance measures analyzed include accuracy percentages, return on investment ratios, as well as capital gains/losses. From this analysis, a general network model can be decided upon to forecast the U.S. 30-Year Treasury bond yield.

# CHAPTER I
# INTRODUCTION TO THESIS

Pattern recognition is a sub-topic of machine learning. The goal of pattern recognition is to classify data based on prior knowledge or statistical information. These data patterns usually consist of several measurements and/or observation values. A complete design of a pattern recognition system consists of gathering information, computing symbolic quantities from that information and classifying the results based on specific performance measures.

The idea of providing technological support in finance has been approached from all directions including support vector machines, logical reasoning, probability, and artificial neural networks. To assist people in making trend forecasts for particular markets, there are more than 250 computer programs in use but these are still limited in ability [12]. Although there has been extensive research for this area, it has not had much impact on financial trend forecasting because of conflicting results, high operation costs and generality issues.

The primary purpose of this study is to analyze the use of pattern recognition systems as an approach to financial trend forecasting. This study consists of designing and implementing three different learning methods: K-means clustering, support vector machines and radial basis functions. The results of each network will be compared to the others to determine their performance of percentage of accuracy, return on investment and capital gains.

## Motivations

Data mining is form of representing and processing knowledge. Pattern recognition falls under the huge umbrella of data mining and allows humans to process the represented data in order to obtain important information – this is done by classification. It is very likely that classification of original data is impossible or very difficult. The use of pattern recognition can potentially improve the methods of selecting stocks and bonds, as well as buying and selling appropriately; that is, if a general network model can be achieved. Lots of research has been completed in this area of financial forecasting but not one general model has been discovered. To achieve this goal it is important to focus on the following questions:

- Can a network be trained to perform the classification desired?
- What is the ambiguity in the problem that makes 100% accuracy impossible?
- Which network structure is appropriate in real-time?

This work aims to further research in supervised and unsupervised learning methods. A forecasting system was designed for the three learning methods to predict weekly swings in U.S. 30-Year Treasury bond yield markets. Because the bond market is also a reflection of the market as a whole, many investors and financial analysts use bond yields as guides to determine whether a commodity should be held, bought, or sold. In other words, if the rate of change of the U.S. 30-Year Treasury Bond is less than a benchmark percentage, say 10%, then the market is doing pretty good. Likewise with a rate of change greater than that percentage, the market is not stable and may not be good for investing [1].

## Organization of Thesis

Chapter II provides a general literature review of similar applications in the areas of financial trend forecasting. It discusses the general details of each of the learning methods. Chapter III presents the actual problem, approach and solutions to forecasting the trend of the U.S. 30-Year Treasury Bond yield. The experimental results and analysis discussion are outlined in Chapter IV. And finally Chapter V includes a summary, conclusions and future work recommendations.

# CHAPTER II

# REVIEW OF THE LITERATURE

This chapter reviews the three areas of relevant literature. First is a brief introduction of pattern recognition and the differences between supervised and unsupervised learning in pattern recognition. An introduction of the concepts of using pattern recognition as a means of financial forecasting is also included.

Second, previous efforts in the area of financial forecasting are discussed. This section includes specific examples of using pattern recognition for finance as well as that pertaining specifically to bonds. This section concludes with a discussion of how these financial forecasting systems are limited in scope and how those limitations may have affected results.

Third and lastly, this chapter reviews the three forecasting methods used in this study. After a brief introduction, this section discusses K-means clustering, radial basis functions (RBFs) and support vector machines (SVMs). It also outlines how each method is implemented. The significance of the results of this study is also mentioned.

## Introduction to Pattern Recognition

Pattern recognition is a field that was formally introduced in the 1960s. It covers subjects in the areas of statistics, engineering and artificial intelligence but more recently, applications employing data mining (such as forecasting). Data mining is the practice of storing huge amounts of data and transforming it into useful information for exploration and analysis. A variety of everyday tasks are completed with the help of pattern recognition. For example, the United States Post Office uses a computer system to read handwritten addresses on envelopes. There are also programs which help flag insurance fraud by helping to spot medical providers whose fees may differ from the normal range.

Pattern recognition systems are divided into three groups: statistical, syntactical and neural. Statistical pattern recognition follows statistics rules and theories; it makes use of discriminant analysis, feature distraction and so forth. Syntactical pattern recognition makes inferences concerning grammar while parsing data and text. Neural pattern recognition is a fairly new field that strictly uses artificial neural networks to solve problems. Artificial neural networks are used for fingerprint identification, voice recognition, classification of objects and much more. For the purpose of this study, statistical pattern recognition methods will be employed.

Statistical pattern recognition usually occurs in several stages; this process is further detailed in the application presented in this paper. The first of these stages is called preprocessing or feature extraction. After collecting the data, preprocessing operates on the data for calculations, to reduce dimensionality and to remove insignificant information. It may be necessary to preprocess the data several times before obtaining the most important features. Then pattern classification or clustering is performed, applying various analysis techniques to appropriately assess the results. Figure 1 displays a diagram of a simple pattern recognition system [16].

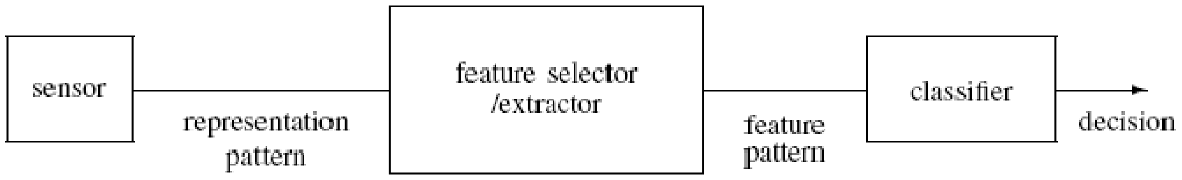


**Figure 1.** System Block Diagram of a Pattern Recognition System

In a pattern recognition system all knowledge is stored as a pattern. The way in which these patterns are processed within the pattern recognition system allows them to model some particular scheme. The most important aspect is to match the pattern to the correct scheme. In most cases of the training phase, every attribute is assigned a 'weight'. This weight is a special calculation mostly obtained from the frequency of occurrence [17].

So in general how is a pattern recognition system designed? First a finite data set is provided. If the data set is too complex there may be too much 'noise', or may exemplify over-fitting (having too many parameters). On the other hand, if the data set is not complex enough then an acceptable pattern cannot be constructed. Achieving optimal performance is important but 100% classification accuracy may not be possible. Choosing the right model and measuring classifier performance are the primary concerns [15].

## Supervised vs. Unsupervised Learning

Using a pattern recognition system in any application involves extracting diagnostic features and making a decision based on the commonalities of the features. The common concern is how the features are used to achieve a desired decision. Any pattern recognition system must use a training method in order for the system to learn and memorize a pattern.

There are several training methods that can be used. Some of the most common pattern recognition models include nearest neighbor methods, discriminant analysis, principal component analysis, cluster analysis, linear programming and genetic algorithms. But all the training methods fall in one of two distinct categories: supervised or unsupervised learning.

Supervised learning involves discrimination, meaning it uses weights and labeled data used as comparisons in the pattern recognition design. Training by supervised learning is by far the most commonly used model in financial prediction tasks. Training begins by supplying the system with input patterns and their actual outputs. The system performs calculations (based on which learning method is used) and generates a predicted output. When these comparisons are done the network "remembers" the differences, or the errors, between the actual and predicted outputs using a form of this equation:

$$\text{error}_i = x_{\text{actual}} - x_{\text{predicted}} \tag{2.1}$$

The weights are then adjusted to minimize the difference between the two outputs. This procedure is repeated until one of two things happens. The training process can end when the error converges to zero (or some desired amount based on the user's specification). Alternative to this solution, the system can also be instructed to stop after a predetermined number of cycles.

Unsupervised learning involves clustering, meaning it consists of unlabeled data and the pattern recognition design seeks to find features that distinguish one group from another. The network generally does not have any desired outputs for comparison. Instead the network is expected to recognize the patterns by having large amounts of input data. The observed input patterns are known as *a priori* information, or prior knowledge. One of the biggest arguments in explaining the advantage of using unsupervised learning is the notion that input has independent causes, and since it actions result from those causes the best representation for an input is in those terms [16]. In other words, if it is known what caused the input then the output can be determined. In Figure 2 an example is provided of how clustering can help separate nonlinear data by mapping to higher dimensions.

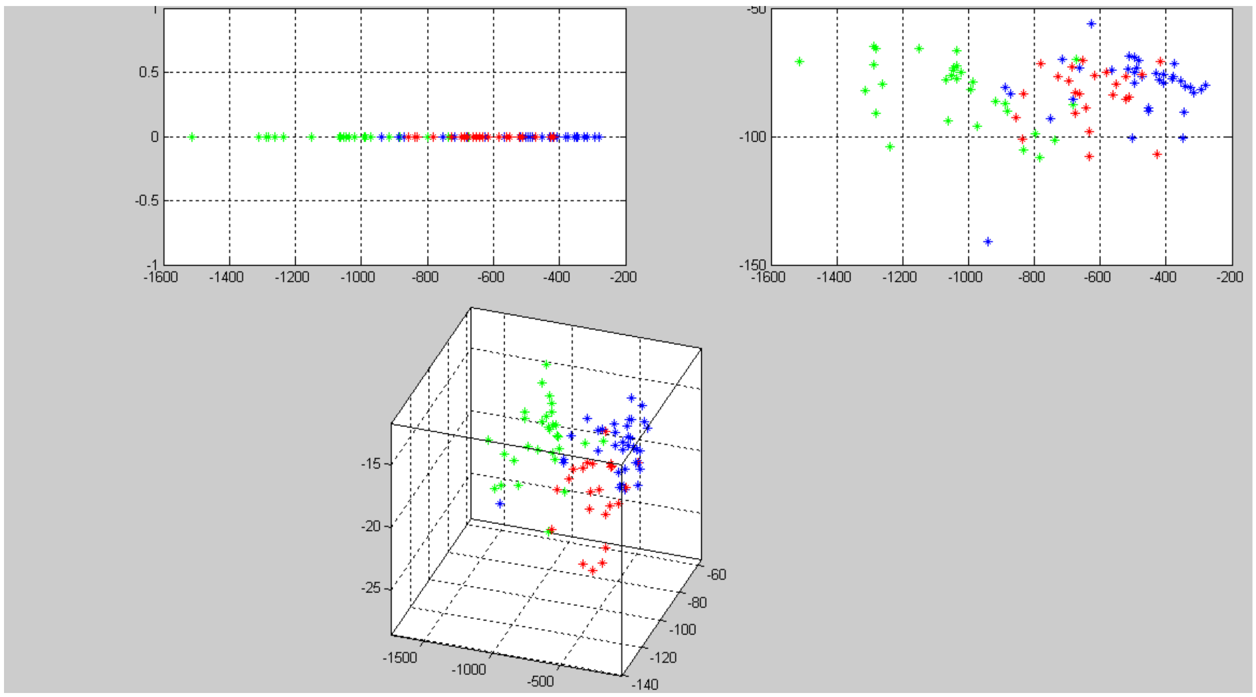


**Figure 2:** 1-D, 2-D and 3-D Representations of Clustering

## Pattern Recognition and Data Mining in Financial Forecasting

In general the approaches to predicting the stock market can be divided into two categories, fundamental analysis and technical analysis. Technical analysis ignores any underlying factors, such as company profit or market sector, and focuses on finding patterns directly from the stock data. Fundamental analysis does just the opposite – it takes all the various factors into consideration. It is widely known that the stock market is a volatile and complex entity which is affected by various factors such as government policies, political situations, public events, internal company politics and much more. However, we have no way of knowing exactly which factor will affect stock prices nor do we know how it will affect the trend of that stock. In the task of financial trend prediction using pattern recognition it is more natural and effective to represent the target values by the successive relative changes in price since the previous time point [1].

There are three ways that any pattern recognition system can forecast a time series: (a) rules relating the current state to the future state are formulated, (b) a fixed set of recent past states are related to future states, (c) the relationship of a large set of past inputs and future states is created [14]. The third method is what is used for this study.

## Previous Efforts in Financial Forecasting

Various solutions to financial market prediction problems have been designed in the past, including but not limited to, support vector machines and RBF networks. When it comes to performing any predictive analysis of a financial commodity it is very difficult to build one model that will fit every market and every security [6]. For that reason there are a variety of solutions available. A discussion of these related previous efforts follows.

### Application Areas

To actually predict the market trend would be beneficial to many investors and analysts who regard fundamental analysis as the best method. For example, the most widely tracked and popular stock index in India is the BSE SENSEX, known for its high liquidity as well as its high level of sensitivity. Dutta et al [6] performed a study of the efficacy of artificial neural networks in modeling the BSE SENSEX weekly closing values and using them to predict the weekly closing values for a subsequent period of time. Feed forward backpropagation networks were chosen because they are able to model non-linear relationships; depending on the level of complexity, hidden layers can be used to achieve a higher level of accuracy. The study found that using a user calculated volatility indicator is an excellent factor to include as an input (as opposed to not using it), producing a mean absolute error of 3.93 percent on the validation set. Market predictions with this small level of error can be quite useful.

Support Vector Machines (SVMs) address the problem that minimization of empirical error does not guarantee small actual error; empirical error is the error produced by the training data, it is described in detail in [9]. Using the Australian Stock Exchange, SVMs were trained for stock selection in an attempt to identify stocks that are likely to outperform the market by having exceptional returns. The equally weighted portfolio formed by the stocks selected by SVMs had a total return of 208% over a five year period, significantly outperforming the benchmark of 71% [9]. The majority of research involving SVMs has found that stock trend movements are predicted significantly well when using only technical (existing data) indicators.

Another popular trend is using pattern recognition in conjunction with natural language processing. Van Bunningen [19] and Falinouss [8] applied SVMs to classify the influence of news articles on the change in stock price. They both hypothesized that it would be better to use an analysis of the combination of news articles and past prices, instead of just past prices, because the market prices are determined by the cause and effect relationship of the two. Van Bunningen performed validations using unique test and data sets for closing prices and news article features; rendering the use of this system impractical because the market is much more random. But the important technique realized was that SVMs are suitable in various forms for financial predictions because they actually extract features that have the most effect on the results. Falinouss used real world time-stamped news articles and intraday prices for the Iran-Khodro Company. His prediction model was able to classify the up or down movement of the company's stock price when a piece of related news is released with 83% accuracy. This is beneficial for corporate investors and analysts to foresee future behaviors.

Other supervised methods that have been explored include amnestic neural networks and linear SVMs. Traditional neural networks do not solve this problem so amnestic neural networks were designed and described in detail in [21]. Amnestic neural networks are an extension of backpropagation; generally speaking the effectiveness of this model depends on present data being more useful than data from long ago. (Old data has less effect on the training result, just like a human gradually forgets events that occurred 10 years ago.)

Unsupervised pattern recognition, particularly using K-means clustering, has not been fully explored. This is because it is difficult to predict behavior patterns from old data. When data is time variant, data selection will influence the training result [21]. Some unsupervised methods that have been explored include radial basis functions and self-organizing maps. iJade (Stock Advisor, Stock Broker, and Stock Analyst) is a series of Java-based programs developed using RBF recurrent networks. The Stock Advisor collects user input and analyzes forecasting responses. Lee [13] tests the system by evaluating parameter selection, long-term and short-term prediction windows and stock prediction performances. It was concluded that the overall percentage error is 1.401%; but the study also demonstrates a feasible solution for online stock selection.

There has been a plethora of research specific to bond performance, mostly using artificial neural networks (ANNs). Compared with stocks, government securities are much more stable and have minimal business risk. In the bond-rating models of Duddy & Shekjar (1988) and Surkan & Singleton (1992), ANNs were compared with standard regression techniques and found to be significantly more accurate [12]. Cheng et al used ANNs to predict similar results, obtaining 67% prediction accuracy [7]. Chaveesuk et al varied their analysis and used radial basis functions to assign the rating of U.S. corporate bonds [5]. The results were significantly lower than the previously mentioned, achieving a maximum of 30.7% accuracy. Castellani and Dos Santos compared the results of the monthly yield of U.S. 10-year Treasury Bonds using multi-layer perceptrons (supervised), fuzzy logic and self-organizing maps (unsupervised) vs. classical modeling approaches (ARIMA). The MLP and SOM networks outperformed the ARIMA models; however, the group concluded that pure data-driven tests may not encompass all the changes that the stock market endures. [21]

## Limitations

There has always been skepticism regarding the usage of computers, or artificial intelligence, in any area. The main concern is the limitation of using a computer, where a computer cannot think nor rationalize for itself. In addition any incorrect data could corrupt results. For each time series of data, there are many alternative methods that can be used for forecasting. These limitations weigh heavy while considering intelligent-based market trend prediction but on the other hand such limitations can also apply to nonintelligent-based tools.

It almost impossible to find a forecasting model that will be best for any and every period, or any and every security. Financial analysts have no real knowledge of what will happen in the real world market either. Traditionally a set of rules are extracted from the patterns of data, then contradictions must be identified. What if these rules are only a small set of the many combinations? Can these rules be applied to any and all cases? There are and will always be problems in implementing technology in the financial field. There is always the possibility that new combinations will be found which may contradict earlier rules; rendering it impossible to confirm that rules are consistent [2].

In order for a pattern recognition system to be implemented there are other problems to consider as well. Standardization, portability and interpretation are three common concerns. A standardization method needs to be applied to the available data, in order to weed out the skewed data. This is where the different types of analysis are important, such as linear discriminant analysis or principal discriminant analysis; a means of normalizing the data in some way is necessary. It must also be possible for each financial establishment, company or geographic region to use the same general form of software. Last but not least, the interpretation of the data determines the prediction. There can be little to no variance between a corporate finance analyst and an everyday investor.

## Description of Methods

The comparison of the three learning methods performed in this study borrows ideas from the previous efforts as well as new innovations. Out of all the research done in combination of time series forecasting, it seems that none has really dealt with taking a large amount of *a priori* information and predicting a short-term trend. This section describes each of the three methods and how they are used to accomplish the task.

### K-means Clustering

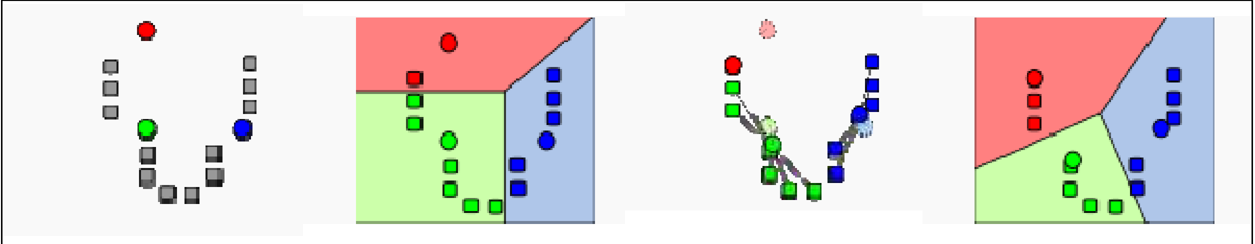
K-means clustering is an unsupervised learning algorithm which is a simple way of classifying a data set through a certain number of clusters. The algorithm was invented in 1956, at that time it was more commonly known as Lloyd's algorithm (although the two are separate entities).

After the data has been reduced to the principal components, it is possible to use K-means clustering to classify. The steps are as follows:

1) Initialize the centroids, which should be equal to the number of classes (targets) in the data set
2) Determine the distance from each object to each of the centroids
3) Select the minimum distance and group each object based on the minimum distance
4) Continue the previous steps until none of the data points are reassigned

Figure 3 demonstrates the K-means clustering algorithm for a 2-D case. Since it is unknown which distance metric provides the best classification, the five that will be used are Euclidean, Manhattan, Chebyshev, Minkowski and Canberra. The results from these metrics were compared to determine which performed the best.

Clustering uses some form of component analysis; in this case principal component analysis (hereafter PCA) is used. PCA is used to reduce the number of dimensions of a data set for easier analysis. It involves determining the eigenvalues, also known as the singular value composition (hereafter SVD) of a data set. PCA transforms the input data so that the elements of the vector set will be uncorrelated; in other words, the elements have no effect on each other. This method has the ability to 'select' the components which have the biggest effect on classification.



Step 1: Initial random centroids and points

Step 2: Points are grouped with the nearest centroid

Step 3: Centroids are moved to the centers

Step 4: Steps 2 & 3 are repeated until desired accuracy

**Figure 3.** K-means Clustering Algorithm Demonstration for the 2-D Case

## Support Vector Machines

Support vector machines (hereafter SVMs) are a set of related supervised learning methods used for classification and/or regression. The original model was introduced by Vladimir Vapnik in 1963 [20]. This algorithm is of particular interest because of the ability to classify nonlinear sets of data, complex systems and modeling the unknown. SVMs map input vectors to a higher dimensional space where a hyperplane is constructed. On a straight line, a hyperplane is a point which divides a line into two rays. In 2D, a hyperplane is a line which divides the plane into two half-planes, as shown in Figure 4. In 3D, a hyperplane is an actual plane which divides the space into two half-spaces. SVMs construct two parallel hyperplanes on each side of the hyperplane that separates the data. The third separating hyperplane increases the distance between the original two hyperplanes.

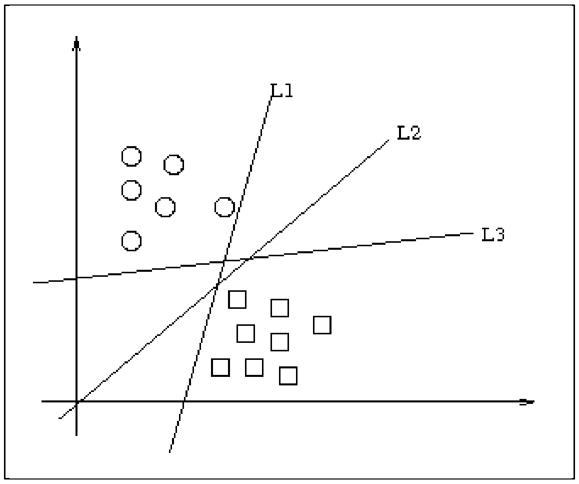


**Figure 4.** Three Hyperplanes Separating Two Classes of Data

As mentioned before, SVMs can be used for classification or regression. In classification it is important to find a hyperplane to separate the classes as much as possible. In regression the goal is find a hyperplane that lays close to as many data points as possible. We will focus on classification in this study. The maximum distance between the hyperplanes will result in a smaller amount of error in classification. The concept is simply stated by Cover's Theorem on the separability of patterns of 1965 [20]:

*"A complex pattern classification problem that is nonlinearly separable in a low dimensional space, is more likely to be linearly separable in a high dimensional space."*

Trafalis and Ince [18] provide a basic description of how SVMs work. Given some training data, a set of points of the vector form **x** and the vector's target class, **c**. The target is usually defined as either 1 or -1 (0) indicating the class to which the point $\mathbf{x}_i$ belongs. Any hyperplane can be written as the set of points, **x**, satisfying

$$\mathbf{w} \cdot \mathbf{x} - b = 0 \tag{2.2}$$

The vector **w** is a vector of weights perpendicular to the hyperplane. The parameter *b* is a bias value that determines the value of the offset of the hyperplane from the origin along the vector, **w**. The weights and bias value should be chosen to maximize the distance between the parallel hyperplanes. These hyperplanes define the data to be classified as

$$\mathbf{w} \cdot \mathbf{x} - b \geq 1, \text{ for Class I} \tag{2.3}$$

and

$$\mathbf{w} \cdot \mathbf{x} - b \leq -1, \text{ for Class II} \tag{2.4}$$

The original model for support vector machines was designed for linear classifiers. Later on Vapnik, along with Bernhard Boser and Isabelle Guyon, suggested a means of non-linear classifiers by applying kernels [20]. The algorithm is fairly similar except that the linear method uses the dot product whereas the nonlinear method uses kernel functions. An activation function can be chosen from the polynomial, Gaussian, sigmoid as well as a variety of others. Choosing the kernel activation function and their parameters are important; some might transform data sets better than others making it easier to classify.

The data set used with SVMs is separated into three parts: training, cross-validation and testing. The training and testing sets are used as with any other network. The cross-validation is a unique data set. Choosing the appropriate kernel function provides good generalization for a particular procedure. SVMs use the cross-validation procedure to ensure that the best kernel activation function is selected on the basis of performance in the training set and not on the basis of performance on external validation sets.

## Radial Basis Function Networks

Radial basis function networks are supervised neural networks which use radial basis functions as the activation functions. They were first introduced by Broomhead and Lowe as early as 1988. The term 'radial' is defined as an object being symmetric around its center; 'basis functions', or kernels, are a set of functions whose linear combination can generate an arbitrary function in a given function space. Basis functions generally used with RBF networks include Gaussian (used for neural networks), multiquadrics (used for various applications) and thin plate splines (mainly used for fitting two variables). These functions are popular because there are very few, if any, restrictions placed on the locations of the points. In other words, RBF networks are great for applications that have data that may not lie on any sort of regular grid [13].

Mostly used in function approximation, radial basis function networks are also multilayer and can be useful in non-linear time series predictions. The general form of a RBF network equation is

$$y(x) = \sum_{i=1}^{N} w_i \varphi(\|x - c_i\|) \qquad (2.5)$$

where $y$ is the output, N is the number of hidden neurons, $w_i$ are the weights, $\mathbf{x}$ is the input vector, $\mathbf{c}_i$ are the center vectors and $\varphi$ is the actual radial basis function that is used. The hidden units are based on the distance between the input vectors and some target vector.

The goal of a radial basis function network is to approximate any continuous function (i.e. sine, cosine) by finding the Euclidean distance from a center point (or another relative point); the precision of the approximation is based on the number of hidden neurons of the network. These centers are chosen to match the distribution of the input training data. Hidden neurons are used to adjust the weights from the inputs to result in a smaller amount of error. Finally the RBF network provides a smooth interpolation of scattered data of any dimension [11] similar to the one shown in Figure 5.

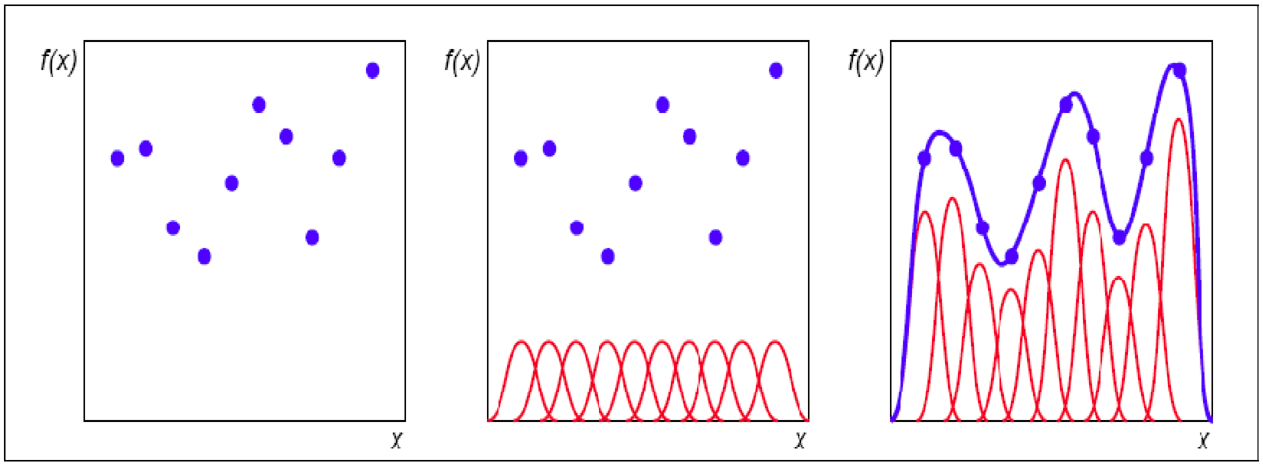


**Figure 5.** Smooth Interpolation of Scattered Data Using RBFs

Although it seems that a RBF network can approximate large amounts of data, there are several disadvantages for this network. First, the network can require lots of memory and can be time consuming to run. Second, a subset of the inputs can be used however a comparison with this subset would be difficult because of the inequality of data. Lastly, smooth interpolation does not result in high predictive accuracy but prediction is slightly better when using a two layer network as opposed to one layer [11], as shown in Figure 6. A one layer network would only use the radial basis function to find the weights. In Figure 6 the two layer RBF network consists of determining the centers first and then use the centers to find the weights (using another neural network approach, like batch-learning).

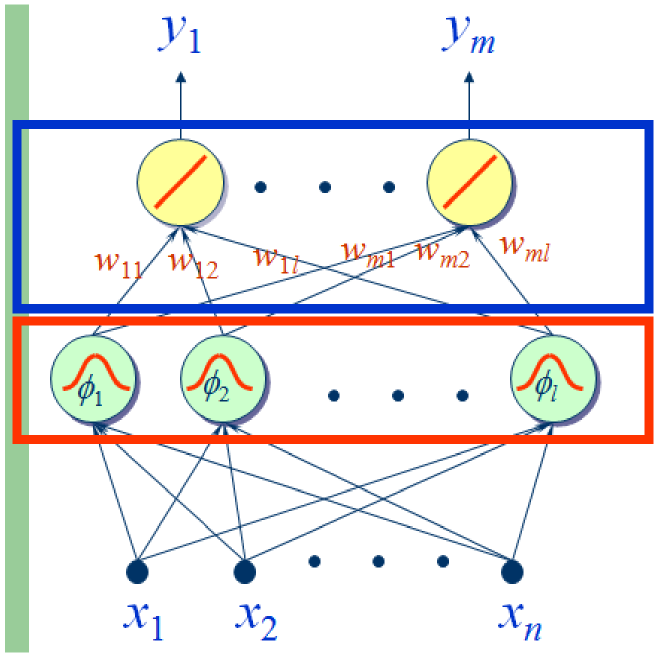


**Figure 6.** Two Layer implementation of an RBF network

## Performance Measures

Accuracy is very important in measuring performance of pattern recognition systems. Since it has already been found that these three networks produce desired accuracy results in some situations, the objective is to now determine which forecasts better for the U.S. 30-Year Treasury Bond yield.

The most challenging task is to fuse information from various external sources (in this case – financial data) with internal sources (calculated measurements and patterns) to make a sound decision. When considering forecasting of any kind, the most common accuracy measurement is the minimization of the mean square error (MSE). MSE is a statistical measurement that quantifies the amount by which an estimator differs from the true value; it is not the actual error. The general formula is

$$\mathrm{MSE(X)} = \left(\frac{\sigma}{\sqrt{n}}\right)^2 \qquad (2.6)$$

where $X$ denotes the entire data set, $\sigma^2$ is the variance, and $n$ is the number of data points in the data set. All the applications mentioned previously used a form of the MSE measurement (mean error or root mean square errors which are deviations of MSE) as the performance measure of accuracy.

In addition to MSE, it is important to assess this system's outputs in terms of finance as well. There are several measurements that can be suited for this task, including return on investment (hereafter ROI), capital gains/losses. The reason it is important to consider these values is because it is more practical to the financial application and can provide more indepth information; for example, the percentage accuracy for a network predicting the change in direction may be small but the ROI value could be very high, and therefore acceptable to an investor looking to increase profit. ROI is a performance measure of the efficiency of an investment whereas capital gains/losses calculations are the measure of actual profit/loss. The ROI is expressed as a ratio or percentage; capital gains/losses are expressed in dollar amounts (but can also be converted to a ratio).

$$\textbf{ROI} = \frac{(\text{Gain from Investment} - \text{Cost of Investment}) + \text{Dividends}}{\text{Cost of Investment}} \quad (2.7)$$

and

$$\textbf{Capital Gain (+) or Loss (-)}: \text{Sale Price} - \text{Purchase Price} \quad (2.8)$$

## Significance of Financial Forecasting

Predictability of market return has been a popular subject for quite some time; there are boggling questions of whether better-than-random predictions of the market are possible. If an investor has forecasting tools, then he/she should be able to increase their profit. But which tool is best? Of all the research that has been reviewed, it seems that only one solution was implemented against a set of data. Each of these networks is different as far as inputs, structure, training method, and so forth. With many data mining models, choosing one technique can be difficult and sometimes involves trial and error. For obtaining a general model, there needs to be a standard technique that always produces acceptable results. The goal of this study was to provide a step in the direction of identifying a standard technique for forecasting the U.S. 30-Year Treasury Bond.

The objective was to use only the properties from the financial time series of 10 years to predict the trend direction of the U.S. 30-Year Treasury Bond yield. Using only historically available data is beneficial in a sense that in real-time, investment decisions can be made without any subsequent foresight. The choice of U.S. Treasury Bond yields is made because of the role that the U.S. bond market plays in the world economy – particularly Europe and Japan [4].

Reducing it to simplest terms, this study incorporates methods of trial and error. Chapter III will present the approach methodology for the three network methods utilized.

# CHAPTER III

# IMPLEMENTATION METHODOLOGY FOR FORECASTING SYSTEMS

Chapter II reviewed the research that has already been using various pattern recognition solutions. Most of the results have helped tremendously in making progress. The success of these solutions has gathered much attention and cause for further research and development.

This chapter presents the development of each of the three methods for predicting the U.S. 30-Year Treasury Bond market trend. The objectives of this study were: first, to use the forecasting results to ultimately state which network is best for this application and thus, secondly, providing a future model for finding the best network for other applications.

The discussion in this chapter covers a summary of the system, including data selection and preprocessing, the actual structures used and finally, the method of analysis of the outputs.

## Introduction

Each network developed in this chapter is designed to forecast the weekly up or down trend of the yield of 30-year U.S. Treasury Bonds. The goal is to predict whether the following Friday’s closing price (and yield) will increase or decrease based on the data available on the previous Friday’s closing. Table 1 illustrates how the results will be considered. The profit cells indicate the forecast was accurate in predicting market direction, whereas the loss cells indicate the wrong prediction. Of course, staying within the profit cells are the desired outputs.

**Table 1.** Classification of Forecasting Results

| **FORECAST** | **MARKET DIRECTION** | |
|---|---|---|
| | **UP** | **DOWN** |
| **BUY** | Profit | Loss |
| **SELL** | Loss | Profit |

Mentioned briefly in Chapter II, bonds are U.S. government securities and is one of the most popular investments. Because of their mature stability over stocks, bonds are attractive to more conservative investors who are worried of losing money. Bills, notes and bonds are the most popular for long-term investments; 30 years is a long time and most people wouldn’t want to risk such a thing. Also, because these securities are actually supported by the U.S. government, there are no additional factors of influence (i.e., company policies, changes in leadership); this alone would attract many investors who have little to no knowledge of finance.

The following section discusses the data selection and preparation process.

## Data Selection and Preparation

U.S. Treasury Bond prices are affected by government policies and other economic factors, like interest rates, credit ratings, and a variety of other influences. There are still so many things to consider, but which ones are the most important? Since this study is focused on a relevant time frame (weekly), those variables that influence such a time frame would naturally be chosen. But that only narrows the choices so much. It is necessary to choose a number of variables which minimizes processing time but maximizes the desired results. The variables chosen for this study were taken from a similar study done by Wei Cheng [6]. She interviewed several investors and financial firms to find out which variables were most important in bond prices. Table 2 is a list of the 19 features selected for this study (including the target value).

Most pattern recognition systems require a large number of iterations. An iteration is defined as one input vector presented to the network. The data used for this study consist of weekly financial information collected for the years 1977 through 1988. It is initially divided into two sets, one for training and one for testing. The information for the years 1977 to 1987 are used for training (541 iterations, 2/18/1977 – 2/13/1987) and the information for the year 1987-88 is used for testing (52 iterations, 2/20/1987 – 2/19/1988). The training vectors were presented in the actual order that they exist to ensure that the format of the vectors did not affect the weight values.

Of the variables used, ten were obtained from the Economic Research and Data portion of the Federal Reserve System. The rest of the variables (nine) were taken from *Yahoo!* Finance. The Target (Predicted Signal) feature was created to represent the past week’s market trend direction: a “0” means the market decreased and a “1” means the market increased or stayed the same. Although the sources for the data are very reliable there is still room for error. For this reason all data must be checked thoroughly for accuracy. The Federal Reserve and *Yahoo!* were the main sources; but the data was also cross-checked with other financial magazines, journals and historical data lists. Some data outliers represent events in the stock market, such as recessions or crashes, so this data is kept in the model. For some of the bond inputs, data is not available for every week (due to circumstances such as discontinuities or holidays). To minimize the effect of these missing values on the networks, the average value of that particular feature column is assigned.

**Table 2.** List of Forecasting Features

| **LIST OF FEATURES** |
|---|
| 3-month treasury bill rate |
| 6-month treasury bill rate |
| 1-year treasury bill yield |
| 2-year treasury note yield |
| 7-year treasury note yield |
| 10-year treasury bond yield |
| 30-year treasury bond price |
| 30-year treasury bond yield |
| Price of crude oil |
| Price of gold |
| CRB Index |
| Federal Funds Rate |
| DJI Average |
| DJI Utilities Average |
| S&P 500 Index |
| Yen Exchange Rate |
| DM/Dollar Ratio |
| Change in M2 |
| Target (Predicted Signal) |

## Methods Employed

### Method 1: K-means Clustering

This algorithm is implemented solely in MATLAB.

The selection of which data to use is an important step but data preparation is just as crucial. The data needs to be collected, organized into a form that is easily analyzed and preprocessed in order to make sure that there are no corrupt values. Figure 1 showed a very simplistic view of the procedure; data may be transformed several times before a final result is achieved. These transformations include removing irrelevant information, reducing dimensions or sorting.

For K-means clustering it is necessary to scale the values of each data point. With so many different variables the actual values vary extensively; some values are expressed in millions while others are expressed as ratios.

Scaling the values would allow the points to have equal effect. There are many ways to normalize data, often it is normalized between the values of 0 and 1. However the formula that will be used for this study is

$$[(\min, \max)] = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (3.1)$$

where $x_{max}$ is the maximum value in the data set, $x_{min}$ is the minimum value in the data set, and $x$ is the data point in question. This formula normalizes the data within the interval range [5].

Once the data has been normalized, PCA analysis is performed and the user specifies how many components to use (the data was trained using one, two, three, and so forth components)). The MATLAB command is

[U,S,V] = svd(cov(input data)) (3.2)

where *svd* is the singular value decomposition (a type of factorization used for PCA). The SVD matrix contains the resulting energy values a feature contributes to the entire data set. These energy values are represented as eigenvectors and form the diagonal of the matrix, arranged from largest to smallest.

Finally the actual K-means clustering algorithm is used. Initially the cluster centroids were set to the mean of the feature column. Then the distance between each column's data point and the centroid is found. The minimum distance is selected as the new centroid (this is done for each of the five distance metrics used). The distance equations used are:

$$\text{Euclidean:} \quad d = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \quad (3.3)$$

$$\text{Manhattan:} \quad d = \sum_{i=1}^{n} |p_i - q_i| \quad (3.4)$$

$$\text{Chebyshev:} \quad d = \max\left(\sum_{i=1}^{n} |p_i - q_i|\right) \quad (3.5)$$

$$\text{Minkowski:} \quad d = \left(\sum_{i=1}^{n} |p_i - q_i|^{\rho}\right)^{1/\rho},$$

where $\rho$ is the order of the distance (3.6)

$$\text{Canberra:} \quad d = \sum_{i=1}^{n} \frac{|p_i - q_i|}{|p_i| + |q_i|},$$

where $p_i$ is a data point and $q_i$ is the centroid point. (3.7)

This process is continued until the centroids no longer have to be updated. At this point the optimal cluster for that feature has been formed. In addition to the measurements used for all three methods, this MATLAB program also keeps track of the number of training loops necessary. After the training is completed, the testing data set is presented to the network. The finalized centroids from the training phase are the ones used to compute the distance metrics and the specific iteration is classified based on the result.

The MATLAB program calculates the relative error and the accuracy percentage by the following equations:

$$Relative\ Error = \frac{1}{N} \sum_{i=1}^{N} | \frac{y_{predicted} - y_{actual}}{y_{actual}} \quad (3.8)$$

where N= total number of data points (rows), and

$$Accuracy\ Percentage = 100 - Relative\ Error \quad (3.9)$$

## Method 2: Support Vector Machines

There are many open source SVM applications available to the average computer user. The one chosen for this study is mcSVM, also known as Multi-Class SVM. This one was chosen because of its ability to classify several classes; it was originally designed by Anguita et al to classify the Iris dataset. mcSVM is run by Windows DOS prompt so all the files must be saved to the same folder on a computer's hard drive. These folders are named 'src', 'dataset' and 'util'; there are also three configuration files that need to be saved.

Again the data and application parameters must be set up. The training and testing data sets must be saved to a text file (Notepad, WordPad). The class of the data should be the last column and all the features should be separated by spaces. An example is provided in Figure 7. Notice that the target value must be included because SVMs are a supervised learning method.

```
8.12  -1.23 3.0   1
 9.08  1.48 1.0   1
 7.44 -6.06 2.0   1
 9.72  4.56 2.0   -1
 7.55  2.20 1.0   1
```

**Figure 7.** mcSVM Training/Testing Set Text File

Both the training and testing files can be named as desired but must be saved within the *dataset* folder in the program's directory. The parameters for the data must be configured accordingly. The configuration file has *.cfg* as an extension and can be modified in any text application. An example of the parameters is provided in Figure 8. The bold characters indicate the values

```
   # fDATA              Dataset file      (dataset/bondtraining.txt)
   # nPATTERN                 N. of patterns    (541)
   # nFEATURE                 N. of features    (19)
   # tVAL                           Validation (T = yes, F = no)  (T)
   # fVALDATA                       Validation data file
     (dataset/bondtesting.txt)
   # nVALPATTERN              N. of validation patterns     (52)
   # tVALTARGET                     Targets for Validation file (T =
yes, F = no)      (T)
   # tNORM                          Normalization     (1)
   #                                      (0 = none)
   #                                      (1 = [min,max])
   #                                      (2 = [mean-
n*stdev,mean+n*stdev])
   #                                      (3 = 2 + saturation)
   # tNN                            n for tNORM (n)
   # tRAND                          Randomize patterns in data file
     (F)
   #                                      (T = yes, F = no)
   # KERNEL                         Kernel type
   #                                      (0 = linear)
   #                                      (1 = gaussian)
   #                                      (2 = normalized polynomial)
```

**Figure 8.** mcSVM .cfg File

used for this study. The same normalization was used for SVMs as that implemented in the K-means MATLAB program ([min,max]). Also, each of the kernel types was used in order to make a broad comparison.

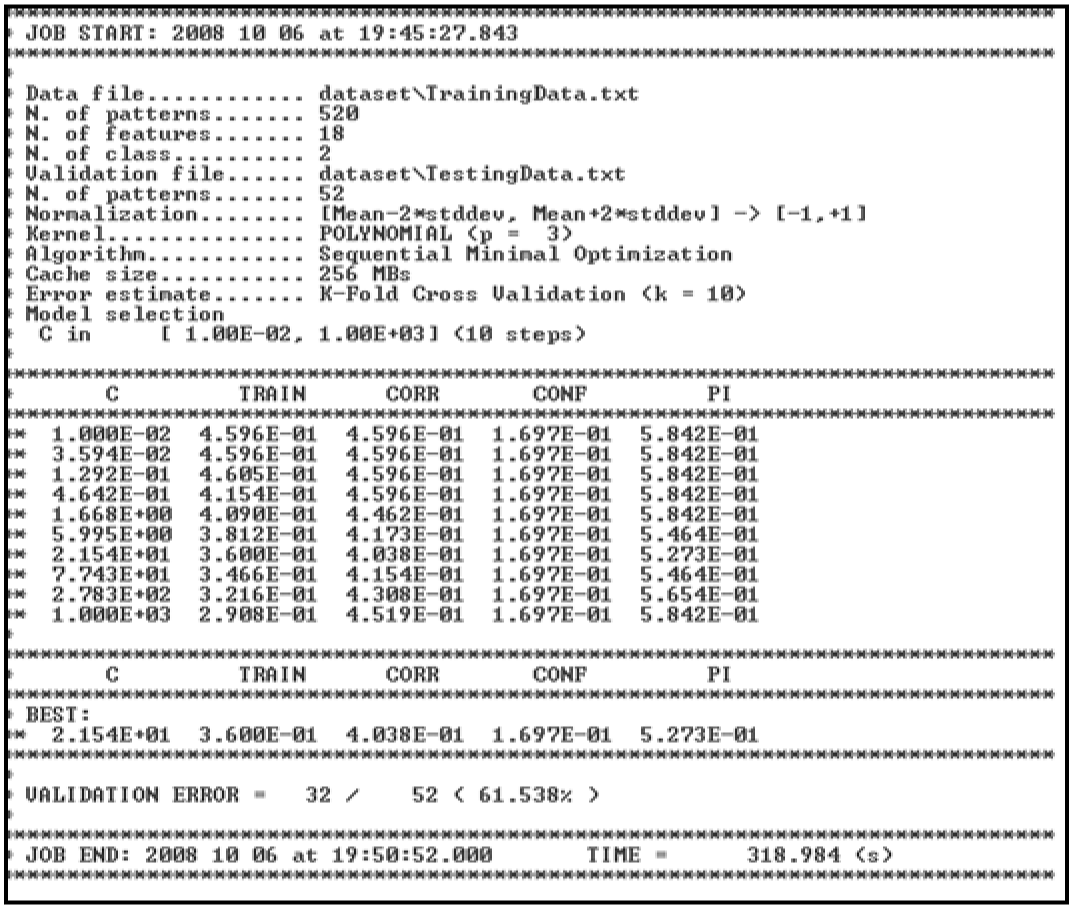
The mcSVM program uses the cross-validation method of ten steps. This is very similar to the number of training loops but not the exact same thing. For this reason, that value will be considered as having no correlation to the performance of the network.

To invoke the mcSVM program, navigate to the folder where the program is saved using the command prompt window. Type the command:

```
mcSVM file
```

where *file* is the name of the .cfg text file created earlier. Depending on the size of the data set it may take a while to run the program. After successful training, mcSVM outputs the results to a *.val* text file. An example of the program's actual output is shown in Figure 9.

The mcSVM calculates the accuracy percentage as well as provides the outputs to show which values were predicted correctly and incorrectly. It also provides the user with the number of training loops necessary.



```
*******************************************************************
* JOB START: 2008 10 06 at 19:45:27.843
*******************************************************************
*
* Data file............ dataset\TrainingData.txt
* N. of patterns....... 520
* N. of features....... 18
* N. of class.......... 2
* Validation file...... dataset\TestingData.txt
* N. of patterns....... 52
* Normalization........ [Mean-2*stddev, Mean+2*stddev] -> [-1,+1]
* Kernel............... POLYNOMIAL (p =  3)
* Algorithm............ Sequential Minimal Optimization
* Cache size........... 256 MBs
* Error estimate....... K-Fold Cross Validation (k = 10)
* Model selection
*  C in     [ 1.00E-02, 1.00E+03] (10 steps)
*
*******************************************************************
*       C          TRAIN      CORR       CONF         PI
*******************************************************************
**  1.000E-02  4.596E-01  4.596E-01  1.697E-01  5.842E-01
**  3.594E-02  4.596E-01  4.596E-01  1.697E-01  5.842E-01
**  1.292E-01  4.605E-01  4.596E-01  1.697E-01  5.842E-01
**  4.642E-01  4.154E-01  4.596E-01  1.697E-01  5.842E-01
**  1.668E+00  4.090E-01  4.462E-01  1.697E-01  5.842E-01
**  5.995E+00  3.812E-01  4.173E-01  1.697E-01  5.464E-01
**  2.154E+01  3.600E-01  4.038E-01  1.697E-01  5.273E-01
**  7.743E+01  3.466E-01  4.154E-01  1.697E-01  5.464E-01
**  2.783E+02  3.216E-01  4.308E-01  1.697E-01  5.654E-01
**  1.000E+03  2.908E-01  4.519E-01  1.697E-01  5.842E-01
*
*******************************************************************
*       C          TRAIN      CORR       CONF         PI
*******************************************************************
* BEST:
**  2.154E+01  3.600E-01  4.038E-01  1.697E-01  5.273E-01
*******************************************************************
*
* VALIDATION ERROR =    32 /    52 ( 61.538% )
*
*******************************************************************
* JOB END: 2008 10 06 at 19:50:52.000       TIME =      318.984 (s)
*******************************************************************
```

**Figure 9.** mcSVM Command Prompt Output

## Method 3: Radial Basis Functions

This algorithm is implemented in MATLAB as well. The data (input and target) is presented to the system and normalized as with the K-means clustering method. Three different RBF networks are trained in two stages.

The program generates data by sampling the input vectors at equal intervals. The first network is a Gaussian mixture model (Gaussian noise is added to the data set). It is trained using the Expectation-Maximization (hereafter EM) algorithm. The EM algorithm is an iterative technique to estimate unknown values given the values of other correlated data. In this case, the unknown values are the centroids of the RBF network and the correlated data is the centroids of the training data set. The mean and standard deviation of the training data set needs to be calculated first. Then using these two initial values, estimate the expected values of the unknown; re-estimating the two parameters yields a new estimate of the expected value of the unknowns. The estimations are repeated until convergence.

The second stage of training is to determine the weights using hidden layers. The number of units in the hidden layer of this study was not predetermined. Each trial varied the number of hidden layers to note the changes of the outputs. Training the network in two stages is very fast, however, the error values are known to be significantly lower [13].

The second network has thin plate spline activations. The same centroids from the Gaussian model are used again but the second layer weights are recalculated. Thin plate splines are similar to the idea of SVMs: they define a new mapping of any input to a new location. Recall the general form of the RBF equation. The kernel function $\varphi$ would be $r^2 \log r$. Similarly for the third network, $r^4 \log r$ activations are used.

# CHAPTER IV

# RESULTS AND ANALYSIS

This chapter presents and analyzes the experimental results of the forecasting systems. Each network is used to forecast the U.S. 30-Year Treasury Bond yield weekly swings for year 1988, given data for years 1977-87. The performances are evaluated in terms of accuracy percentage, return on investment, and capital gains/losses.

The key questions that are answered by these results are: First, which model forecasts more accurately? Second, which model provides the highest return on investment and/or capital gain? Third, can one model be said to have the best overall function for the task at hand?

## Evaluating the Performance of the Networks

The forecasting results obtained from each network is outlined and discussed in this section.
Figures 10 and 11 present graphs of the U.S. 30-Year Treasury Bond yield and its weekly change, respectively. The graph reflects the period of dates included in the testing data set (2/20/1987 – 2/19/1988). These graphs are images of what we hoped to obtain from each network tested with the data set.

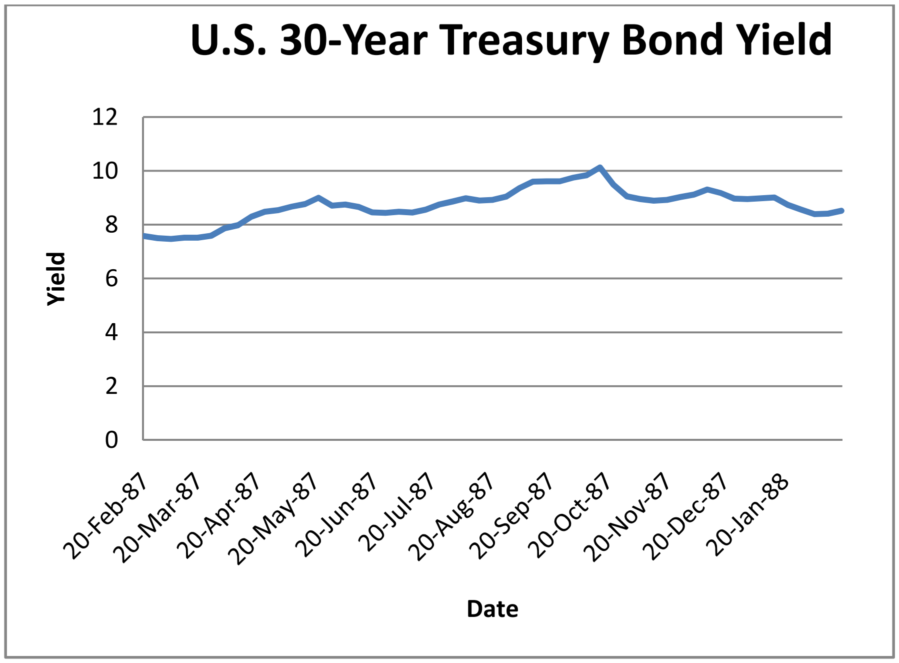


**Figure 10.** U.S. 30-Year Treasury Bond Yield for the Testing Data Set

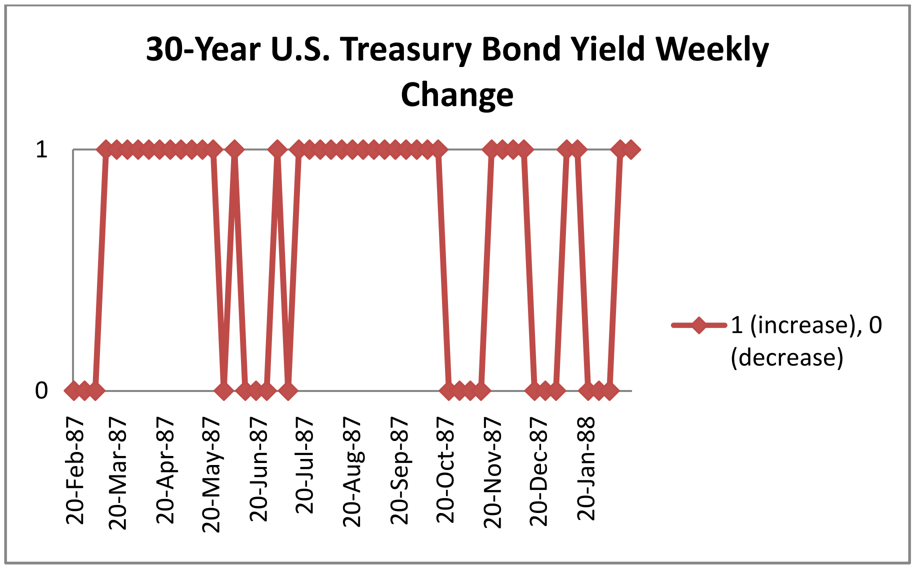


**Figure 11.** Interpretation of Weekly Change for Testing Data Set

To demonstrate the usefulness of PCA, Figure 12 shows the variance of the data. The red tick marks show the variance of the data corresponding to how many components are chosen. Recall that PCA calculates the 'energy', or influence, that a particular number of components contribute to the ability to recognize patterns. Before calculating the result of PCA, the algorithm calculates the covariance matrix (part of SVD). Covariance is a measure of how much the dimensions of a data set vary from the mean with respect to each other. For calculating an n-dimensional data set, the covariance is calculated $\frac{n!}{(n-2)!*2}$ times (Smith, 2002), making up the covariance matrix. The formula for the matrix is

$$C^{mxn} = (c_{i,j}, c_{i,j} = cov(Dim_i, Dim_j)) \tag{4.1}$$

where $Dim_x$ is the $x$th dimension. Basically the formula creates a square matrix and each entry in the matrix is the result of calculating the covariance between two separate dimensions. Finally, from these values the eigenvalues are calculated.

The variance is very high between one and two components, and there is little to no variance after approximately the fifth component. These values become very important when considering accuracy percentage; this graph shows that the network will most likely not change accuracy percentage when using a higher number of components.

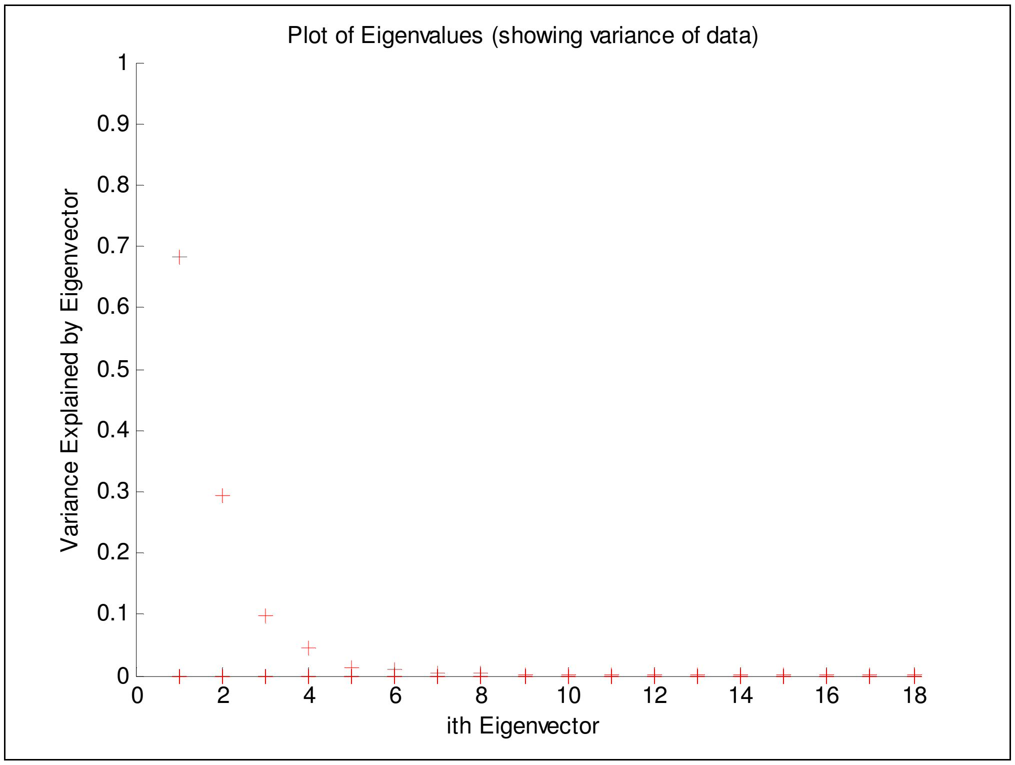


**Figure 12.** Plot of Eigenvalues Showing Variance of Training Data Set

Because the entire PCA algorithm is computed mathematically in MATLAB it is difficult to determine exactly which components correspond to these eigenvectors. Luckily, MATLAB organizes the resulting matrices for the user. Figure 13 displays the capability of the training set being classified. Because four or more dimensions cannot be represented graphically, only up to the first three components are provided. Class I corresponds to the weeks that the market decreased and Class II represents the weeks the market stayed the same or increased.

Although it may not seem to look much more separable as the number of dimensions increases, one must imagine the separability of using all the influential components to incorporate variance. We see from Figure 12 using about three dimensions would give the best separability. Recall that using the corresponding number of components does not guarantee highest accuracy of classification. Figure 13 is only an example of how PCA would work on a data set such as the one used in this study; it is not a true representation of how the data will actually be classified.

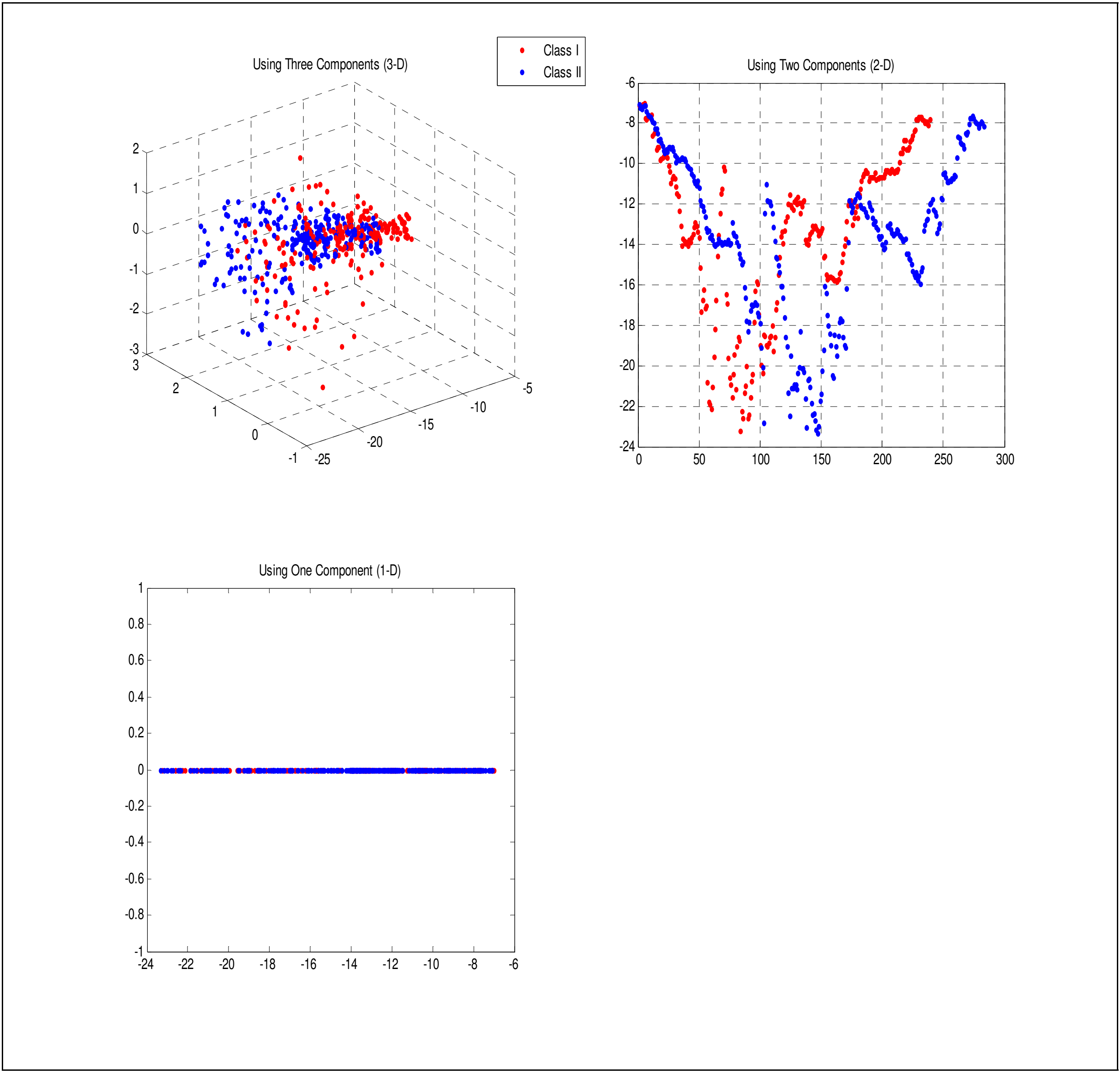


**Figure 13.** Example of Separability of Training Data Set

## Measurement of Percentage Accuracy

Percentage of prediction accuracy is the most common method of tracing network accuracy. The task at hand is to predict whether the bond yield value will increase or decrease from week to week, with respect to the previous week. This performance measurement is fairly easy for the overall movement. Separate analysis was done by the up-and-down weekly movements, however, the same general equation was used.

For K-means clustering, Table 3 shows the highest, lowest and average values of the overall accuracy percentage. It also provides the average accuracies for up and down weekly change movements. It shows that the majority of the networks achieved the same highest accuracy of about 67%. Chebyshev and Minkowski produced the best accuracy for up-weekly-change movements of 70.6% but had the worst accuracy for down-weekly-changes (21.05%). These two equations are fairly similar, hence the striking similarity in their results. Although Canberra had the worst average accuracy, it had the ability to predict down-weekly-change movements very well compared to the rest (83%).

**Table 3.** Accuracy Percentages of K-means Clustering Network

| | **Highest Accuracy Achieved** | **Lowest Accuracy Achieved** | **Average Accuracy Achieved** | **Weekly Change Accuracy: Up** | **Weekly Change Accuracy: Down** |
|---|---|---|---|---|---|
| **Euclidean** | 66.98 | 43.40 | 44.71 | 26.47 | 55.84 |
| **Manhattan** | 66.98 | 40.57 | 42.35 | 16.50 | 74.18 |
| **Chebyshev** | 66.98 | 66.98 | 66.98 | 70.59 | 21.05 |
| **Minkowski** | 66.98 | 66.98 | 66.98 | 70.59 | 21.05 |
| **Canberra** | 52.83 | 35.85 | 46.33 | 36.43 | 83.04 |

Over the entire period (one year) the Chebyshev and Minkowski metrics performed the best out of the five, accurately predicting the trend 67% of the time. These results suggest that K-means clustering can result in significantly different performance when predicting up-weekly-change and down-weekly-change movements. The results support the conclusion that, in general, a K-means clustering network using certain distance metrics can process data for market change in one direction better than that of the other.

Judging from Figure 14, it seems that only the Manhattan and Canberra distance metrics had the most variety in the number of training loops used when the number of components changed. This is in accordance with their performance, shown in Figure 15. Canberra had the lowest average accuracy precisely when its network used only one component; the rest of the time the number of training loops varied while its accuracy seemed to increase and decrease with no particular pattern. The Minkowski distance metric was one of the two that had the highest average accuracy and it also used the most training loops whereas the Chebyshev metric used less than half the number of loops than that of Minkowski. From the two figures, it cannot be easily deduced that the number of training loops directly affects the accuracy of the network but there is some correlation.  Also it can be assumed that training a network using the Chebyshev metric would be less time-consuming and use fewer resources than that of the Minkowski, while still maintaining good results.

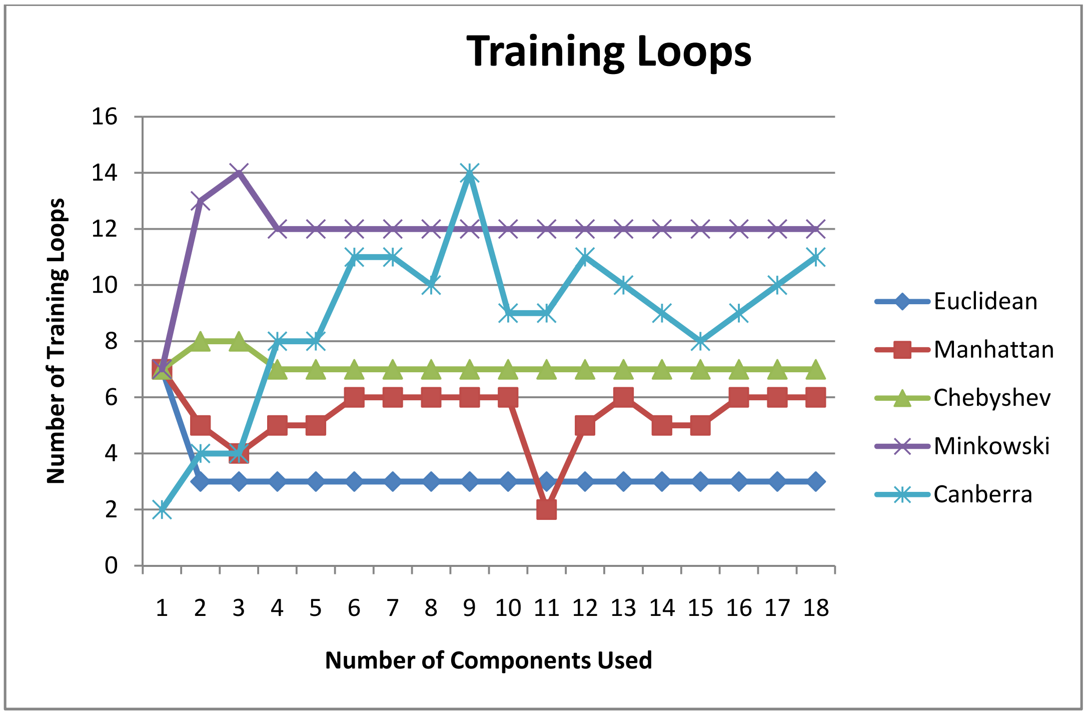


**Figure 14.** Plot of Training Loops for K-means Clustering

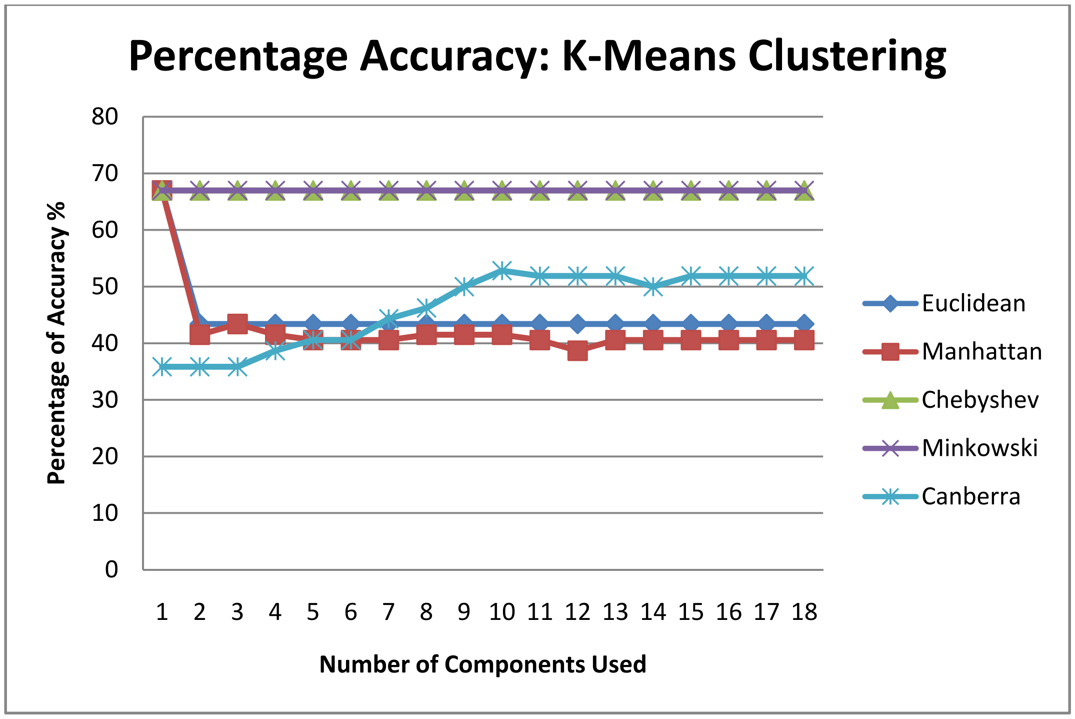


**Figure 15.** Plot of Average Accuracy Percentages for K-means Clustering

The results for SVM percentage accuracy are not as extensive as that of K-means clustering but still present a good visual of how the network performs. Table 4 shows the overall accuracy values as well as the up- and down-weekly change accuracies. There was no need to do multiple numbers of trials for these networks because the results would not change.

**Table 4.** Accuracy Values for mcSVM

| | **Average Accuracy Achieved** | **Weekly Change Accuracy: Up** | **Weekly Change Accuracy: Down** |
|---|---|---|---|
| **Linear** | 65.38 | 100 | 0 |
| **Gaussian** | 69.23 | 97.06 | 15.79 |
| **Poly (deg 2)** | 40.38 | 11.76 | 89.47 |
| **Poly (deg 3)** | 40.38 | 11.76 | 89.47 |
| **Poly (deg 4)** | 34.62 | 0 | 100 |

Again these results support the same conclusion as before that SVM networks using certain parameters can predict market change in one direction better than that of the other. The linear and Gaussian kernel functions (gamma = 1) provide similar results, ranging between 65% and 70% overall accuracy. The polynomial kernel function of degree 4 performed the worst overall but achieved 100% accuracy for the down-weekly-changes. All the polynomial kernel functions predicted the down-weekly-changes very well but not so good for the up-weekly-changes. The Gaussian kernel had the best performance, over the rest of the SVM networks as well as the K-means clustering networks.

The RBF networks did not predict very well at all. There are a limited number of plots displaying the function fitting capability because there were so many dimensions to the data sets of this study. Similarly, because of how the network is structured the up- and down-weekly changes cannot be calculated.

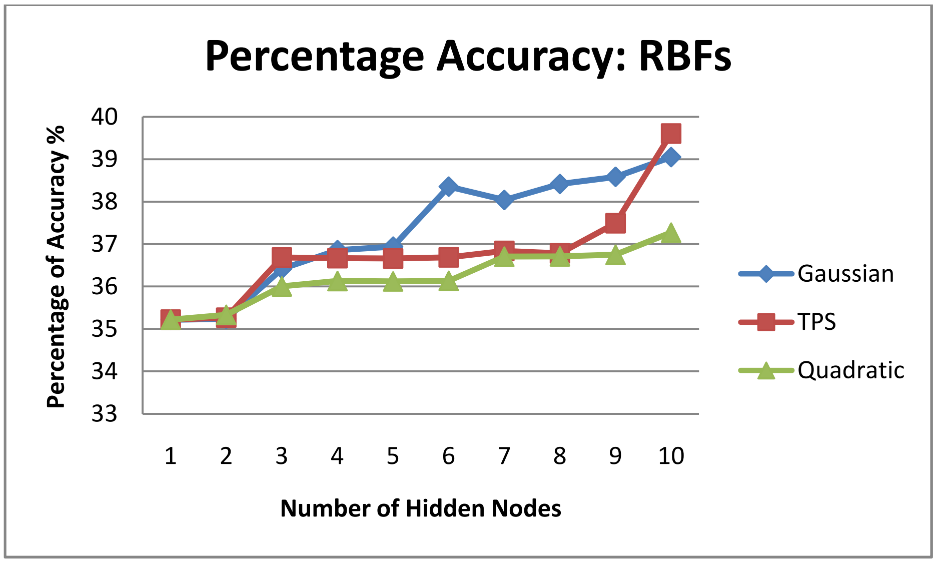


**Figure 16.** Plot of Accuracy Percentages for RBF

The number of hidden nodes was varied with each metric, up to ten. The number ten was chosen because usually the number of hidden nodes is equal to about half of the number of features. Figure 16 displays the results and the average accuracies are shown in Table 5.

**Table 5**. Accuracy Values for RBFs

| | **Highest Accuracy Achieved** | **Lowest Accuracy Achieved** | **Average Accuracy Achieved** |
|---|---|---|---|
| **Gaussian** | 39.05 | 35.74 | 37.31 |
| **Thin Plate Spline** | 39.61 | 35.75 | 36.79 |
| **Quadratic** | 37.27 | 35.84 | 36.24 |

From Figure 16, it can be concluded that the number of hidden nodes has a direct correlation to the accuracy percentage of RBF networks; the more hidden nodes the greater the accuracy. The highest accuracy achieved was that of the Gaussian kernel function; although it is not nearly as high as that of the SVM it still supports the hypothesis that it is the best function for high-dimensional data.

All three metrics produced very similar results right around 37%, about half of the highest accuracy for the other two methods. Figure 17 shows how the data fit the functions mentioned above. Normally when fitting functions the target space varies; yet the unique thing about this study is that the target can only be 0 or 1. Although up- and down-weekly-changes cannot be intuitively analyzed, Figure 18 illustrates that the majority of the accuracy percentage can be credited to those points that satisfy the underlying function.

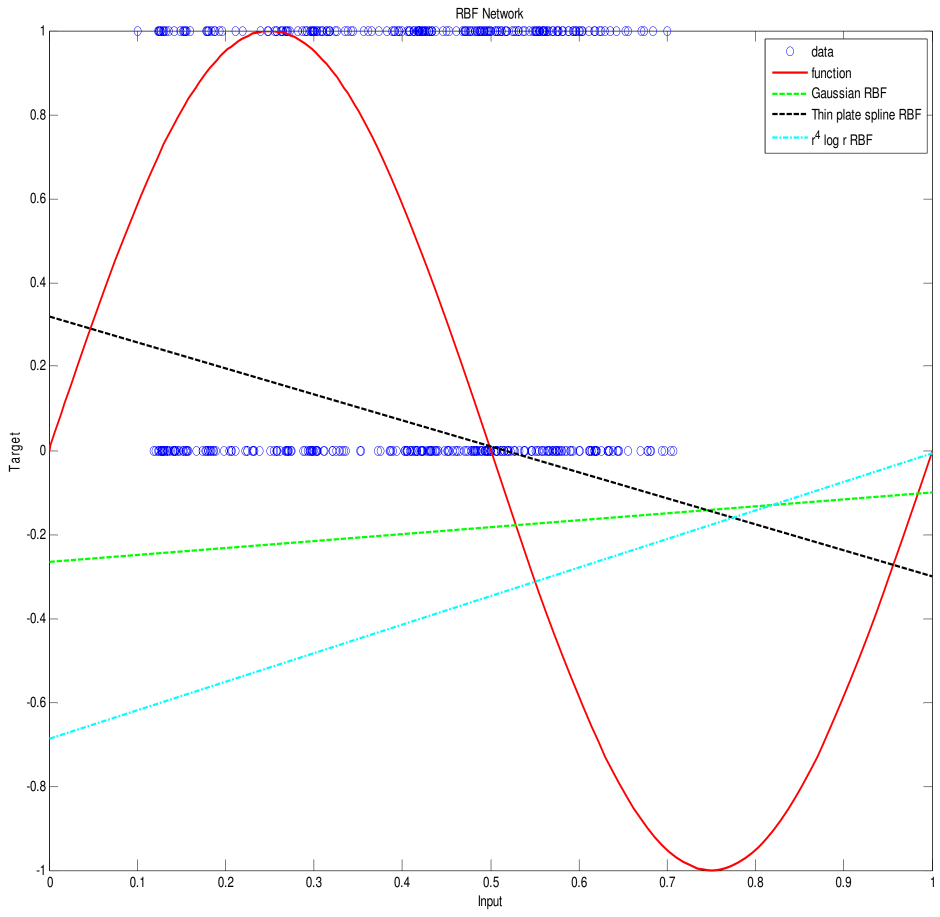


**Figure 17.** Underlying Function of Testing Data Set

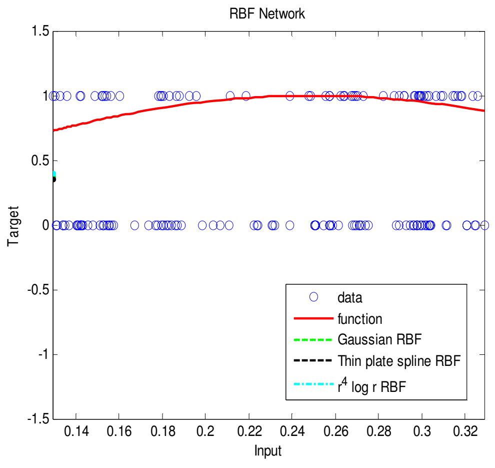


**Figure 18.** Zoom-In on Underlying Function

## Measurement of Return On Investment (ROI)

Although this study is mainly focused on which method can predict the bond yield trend the best, there are still other performance gauges to consider. In the U.S. Treasury Bond market, prices fluctuate from week to week but the amount of fluctuation is not always the same. For example, in the testing data set there were 34 yield increases but only 19 yield decreases. Can it be said that this time period had a rising market? The answer is: not necessarily. In other words, it could be possible that the decreases were greater in amount so there may have only been a very small gain over the period. In this situation the accuracy percentage would not be the best performance measure of the ability of the network to predict the trend.

Table 6 summarizes the performance of ROI for K-means clustering. Keeping in mind that treasury bonds are usually held by investors for a number of years, these ROI values may be smaller than what is expected. Recall that the forecasting systems only predict a year in advance (as opposed to year**s**, in which case the ROI values would be much higher).

Figures 19 - 22 are graphs of each distance metric and their ROI values based on the number of components that were used. It seems that some of the networks ROI percentages are mirrors of each other. This correlation can be attributed to the accuracy percentages because when a network performs well in forecasting one direction then there will be opposite performance for the other direction. The Manhattan metric's ROI fluctuated the most for up-weekly-changes, but also had the best ROI performance overall. The Chebyshev, Minkowski and Euclidean metrics were very stable.

**Table 6.** ROI Performance for K-means Clustering

| | **Average ROI (%)** | **Average Weekly Change ROI: Up (%)** | **Average Weekly Change ROI: Down (%)** |
|---|---|---|---|
| **Euclidean** | 11.48 | -0.81 | 0.75 |
| **Manhattan** | 12.73 | 0.87 | 0.29 |
| **Chebyshev** | 10.73 | -0.01 | 1.09 |
| **Minkowski** | 10.73 | -0.01 | 1.09 |
| **Canberra** | -4.63 | -0.37 | 0.39 |

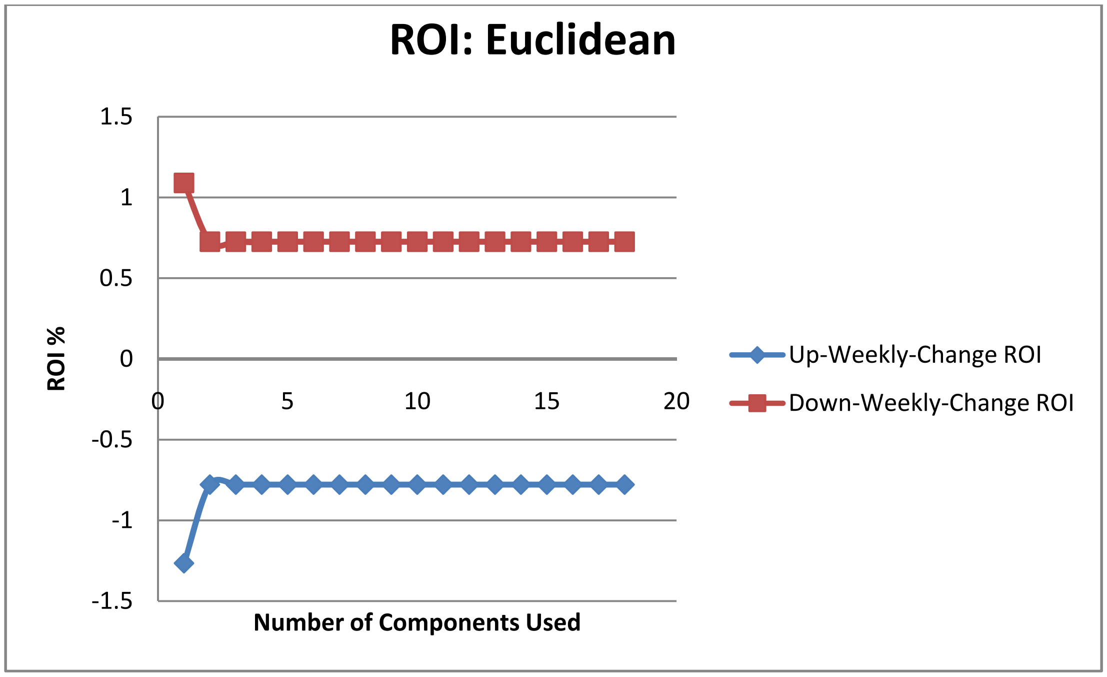


**Figure 19.** Euclidean ROI for Weekly Changes

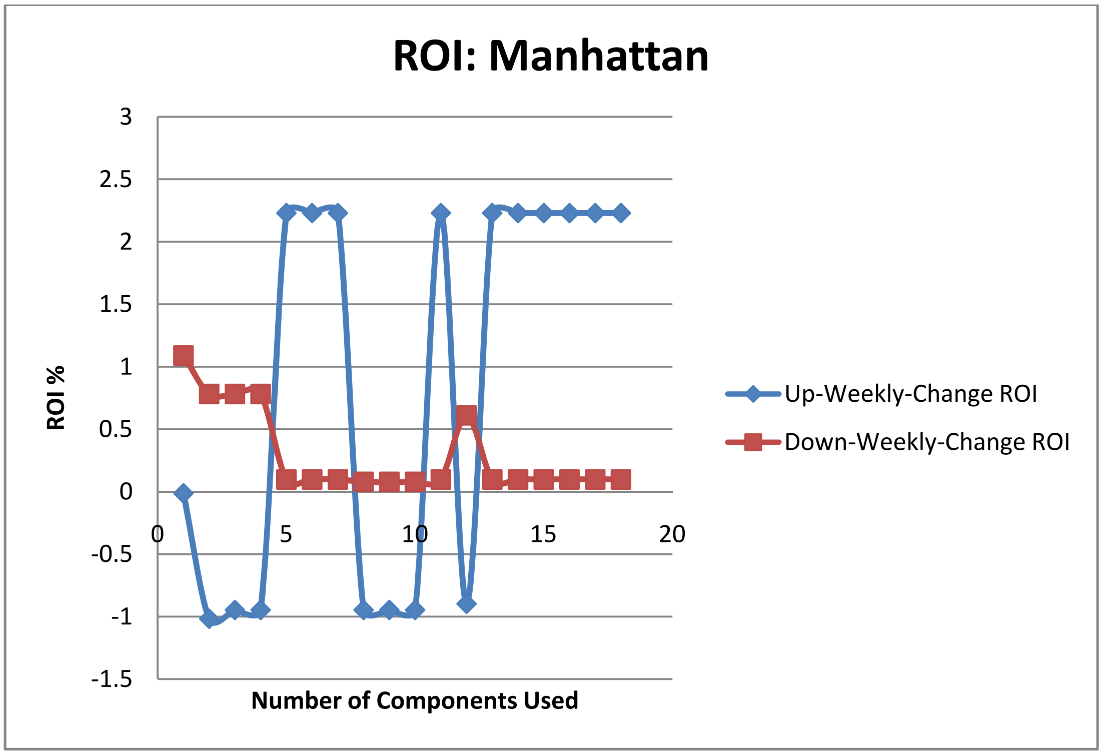


**Figure 20.** Manhattan ROI for Weekly Changes

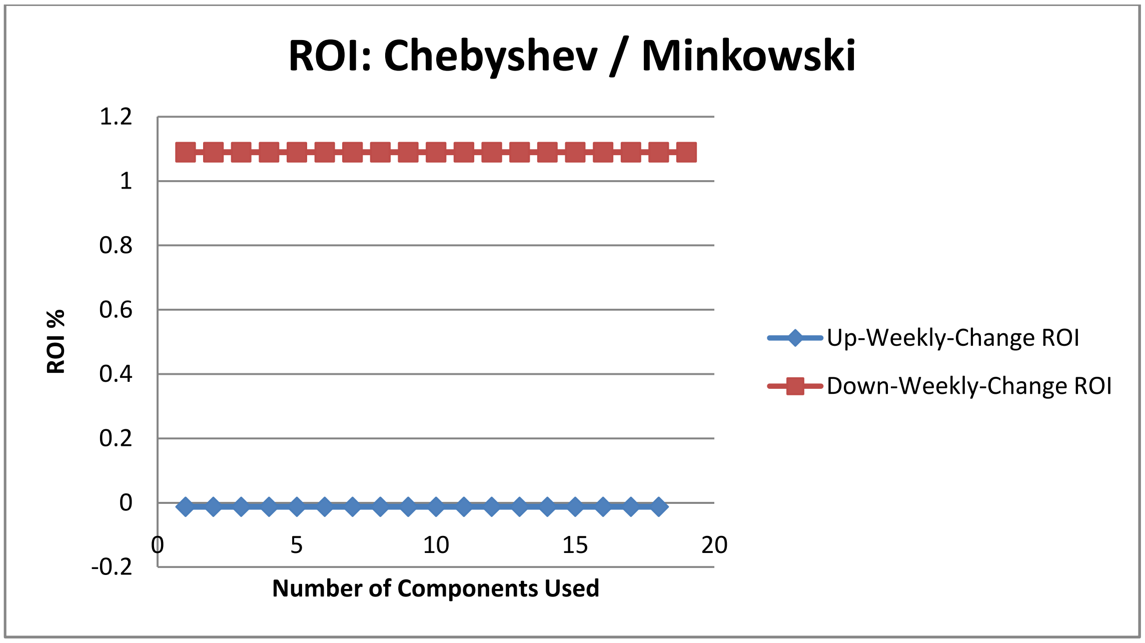


**Figure 21.** Chebyshev and Minkowski ROI for Weekly Changes

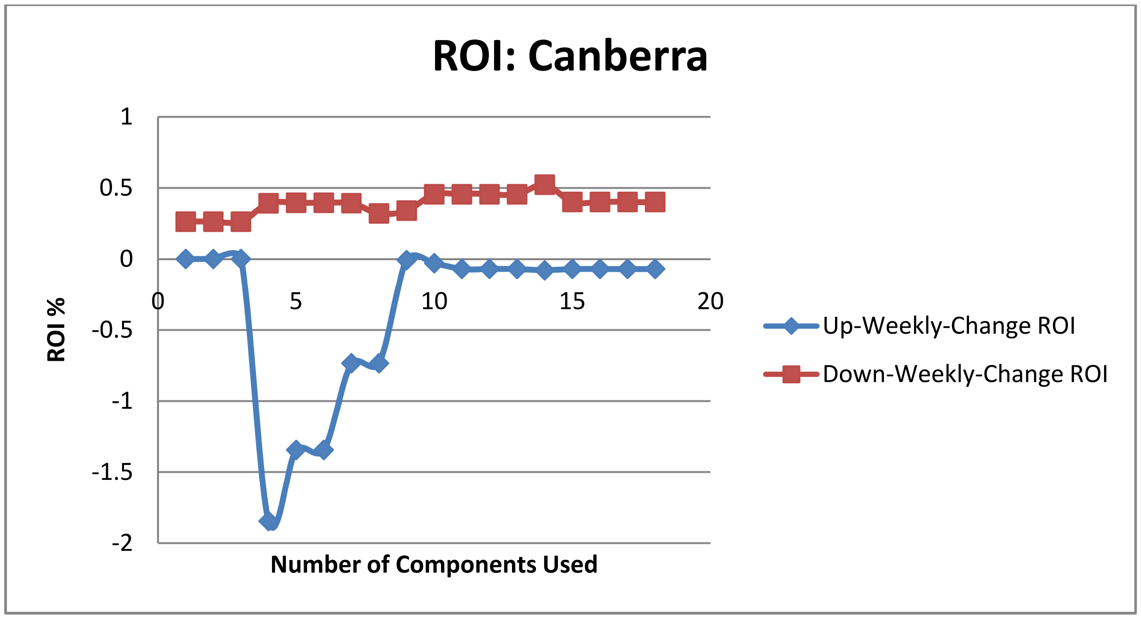


**Figure 22.** Canberra ROI for Weekly Changes

Table 7 gives an overall view of the ROI performance measure for the mcSVM program. Each network showed a profit for predicting the up-weekly-change movements but a loss for the down-weekly-change movements. If it was expected that ROI would move according to percentage accuracy, these values would certainly prove that theory invalid.

**Table 7.** ROI Performance for mcSVM

| | **Overall ROI** | **Weekly Change ROI: Up** | **Weekly Change ROI: Down** |
|---|---|---|---|
| **Linear** | 13.48 | 1.23 | -1.85 |
| **Gaussian** | 11.62 | 1.22 | -1.66 |
| **Poly (deg 2)** | 15.45 | 1.22 | -1.66 |
| **Poly (deg 3)** | 11.62 | 1.22 | -1.56 |
| **Poly (deg 4)** | 15.45 | 1.22 | -1.66 |

Figures 23 and 24 show plots of the relationship between each metric's forecasting accuracy and its ROI. It shows that profitability and accuracy have somewhat of a relationship but not a very strong one. For example, the Manhattan metric was the only one which resulted in a positive ROI for the up-weekly-changes but surprisingly, does not have high percentage accuracy for those. The Canberra metric performed the worst regarding accuracy and also performs the worst for the ROI measure. Alternatively, all the networks showed a profit when forecasting down-weekly changes. Table 3 indicates that the highest percentage accuracy was from Chebyshev and Minkowski but these two metrics did not have the highest ROI values (but still returned a profit). Similarly, the Gaussian network had the highest accuracy value of all the other metrics (including K-means clustering) but had one of the lower ROI percentages.

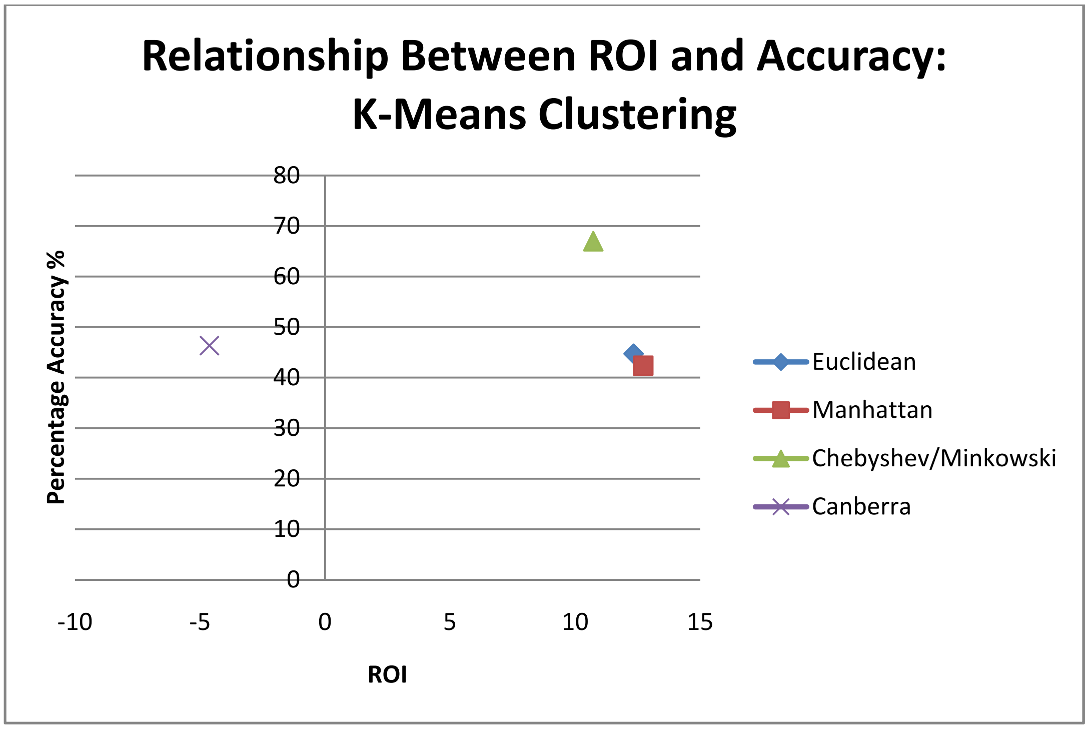


**Figure 23.** Graph of ROI and Accuracy Relationship for K-means Clustering

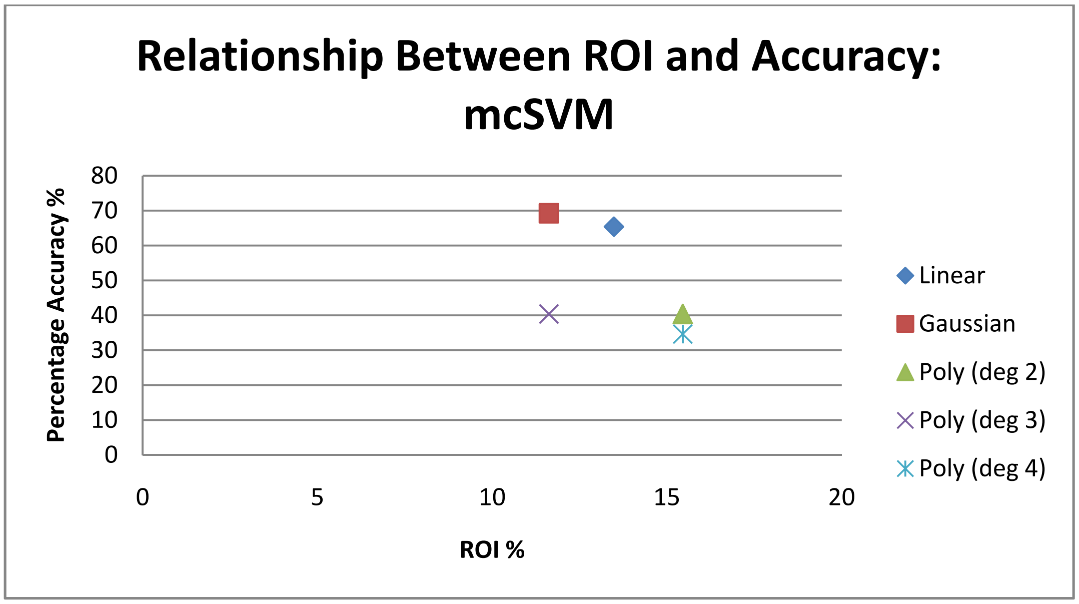


**Figure 24.** Graph of ROI and Accuracy Relationship for mcSVM

Using the Pearson correlation coefficient, the value for the K-means clustering network is 0.223 and the value for the SVM network is -0.94! These two values ensure that a high forecasting accuracy does not guarantee great returns from a particular investment; but it is more likely that profits will increase when using K-means clustering and decrease when using SVMs.

Unfortunately it is not very intuitive which data points were predicted correctly and those which are not for the RBF networks. For this reason the ROI performance measure analysis is not available.

## Capital Gains and Losses

The third factor to consider is financial risk. Profitability is attractive to many investors but normally under a controlled amount of risk. Over a given period of time there may be great profitability but risk for large losses could also be very high. Measuring the loss experience of these models allows the high risk potential model to be exposed. Several kinds of losses and gains are realized in this section: (1) the largest loss from a single trade, (2) the largest cumulative loss in consecutive down-weekly-changes, (3) the largest gain from a single trade, and (4) the largest cumulative gain in consecutive up-weekly-changes. It is also to be analyzed whether there is any correlation between percentage accuracy, ROI and the capital gains/losses.

The summarized results are presented in Table 8 and Table 9.

**Table 8.** Capital Gains/Losses for K-means Clustering

| | **Largest Loss: Single Trade (%)** | **Largest Loss: Consecutive Trades (%)** | **Largest Gain: Single Trade (%)** | **Largest Gain: Consecutive Trades (%)** |
|---|---|---|---|---|
| **Euclidean** | -8.48 | -13.6 | 4.24 | 16.13 |
| **Manhattan** | -4.24 | -13.51 | 4.24 | 16.03 |
| **Chebyshev** | -4.24 | -13.51 | 4.24 | 16.03 |
| **Minkowski** | -4.24 | -13.51 | 4.24 | 16.03 |
| **Canberra** | -4.24 | -19.6 | 4.24 | 10.2 |
| **Summary:** | -5.09 | -14.75 | 4.24 | 11.88 |

**Table 9.** Capital Gains/Losses for mcSVM

| | Largest Loss: Single Trade (%) | Largest Loss: Consecutive Trades (%) | Largest Gain: Single Trade (%) | Largest Gain: Consecutive Trades (%) |
|---|---|---|---|---|
| **Linear** | -14.97 | -4.11 | 6.23 | 22.38 |
| **Gaussian** | -14.97 | -4.11 | 6.23 | 22.38 |
| **Poly (deg 2)** | -6.23 | -19.34 | 4.37 | 17.35 |
| **Poly (deg 3)** | -6.23 | -19.34 | 14.97 | 17.35 |
| **Poly (deg 4)** | -6.23 | -22.38 | 14.97 | 4.11 |
| **Summary:** | -9.73 | -13.86 | 9.35 | 16.71 |

Examining the tables, it is found that the largest loss and the largest gain both occurred with the SVM network using the fourth degree polynomial and linear/Gaussian kernel functions, respectively. All the metrics for K-means clustering had a largest single trade gain of 4.24% and each had moderate gain for consecutive trades as well. The SVM networks witnessed higher gains than that of K-means clustering, both as a single trade as well as consecutive trades. Unlike that of ROI and percentage accuracy, it can be said that capital gains/losses and accuracy are heavily correlated for SVMs. The Pearson correlation coefficient is 0.78 and 0.99 for capital gains and losses (consecutive), respectively; this shows a very strong correlation validating that if percentage accuracy is increased then capital gains will increase as well. Contrastingly, for K-means clustering, the correlation is only 0.34 and 0.33 for largest losses and largest gains (consecutive), respectively.

Regarding ROI and capital gains/losses, the correlations are not very strong. The Pearson correlation coefficient for SVMs is -0.52, stating that a system predicting positive ROI does not guarantee high capital gains. The opposite is true for K-means clustering: positive ROI will most likely result in moderate capital gains.

Unfortunately, as stated before, it is not very intuitive which data points were predicted correctly and those which are not for the RBF networks. For this reason the capital gains/losses performance measure analysis is not available.

## Discussion of Results

Several conclusions can be made from the performance measures explained:

1. The results presented in this section show that there are both supervised and unsupervised methods have strong and weak aspects for forecasting U.S. long-term bond prices.
2. The results showed that there is a small difference between the performance of a supervised network (namely, SVMs) and an unsupervised network (K-means clustering), but each network will witness different performance for up- and down-weekly changes.
3. For an application such as long-term bond forecasting, radial basis function networks cannot be properly analyzed and also do not perform as well as other learning methods.
4. The results provide evidence to conclude: (1) the data set can be classified when using higher dimensions; (2) both supervised and unsupervised learning methods can achieve close to the same overall percentage accuracy; (3) some networks predict weekly better in one direction than the other.
5. High percentage accuracy is correlated to capital gains/losses for supervised methods but uncorrelated for unsupervised methods. ROI is not correlated to percentage accuracy or capital gains/losses.
6. In both cases of the SVM and K-means network, the metrics which performed the best were able to accurately predict the up-weekly-changes better than the down-weekly changes. It is also true that the metrics performing the worst overall could predict the down-weekly-changes very well.

# CHAPTER V

# CONCLUSIONS AND FUTURE WORK

The goal of this study is to determine if supervised or unsupervised learning methods are suitable for a particular financial forecasting application. After all three networks were developed, they were tested to determine which could forecast accurately and/or produce the most profit.

This chapter summarizes what was found. The results and conclusions are discussed further. Finally, this chapter includes contributions of this study as well as suggestions for future work.

## Overview

To reiterate, the study has several objectives:

1. To analyze the use of a pattern recognition system as an appropriate approach to financial forecasting.
2. To determine whether supervised or unsupervised learning methods should be used in regards to variable data (i.e., stocks or bonds market).
3. To develop a general model for forecasting the U.S. long-term bond.

Chapter II presented a literature review of the three network methods for time-series forecasting. It discussed specific previous applications in the area, and also described the strengths and weaknesses of each. A background of each method was also outlined; advantages and disadvantages of each were also discussed. Finally, the performance analysis process was discussed.

Chapter III described the development of each network method for forecasting the U.S. 30-Year Treasury Bond market. This includes how the data is selected and processed, training, network formation and evaluation criteria.

Chapter IV discusses the results of each system. They were each evaluated by forecasting the long-bond market for the period, 2/20/1987 – 2/19/1988. The effectiveness of each network was assessed for percentage accuracy, return on investment and capital gains/losses.

## Further Conclusions

The three pattern recognition systems were successfully implemented. The performance of the K-means clustering and SVM networks shows that supervised and unsupervised learning methods can be useful for forecasting financial markets. On the other hand, it seems that RBF networks are very hard to train using so many dimensions. If one factor totally influenced the bond market then RBF networks would be a good choice. However, judging from the results presented in Chapter IV RBFs is clearly not the right choice for the U.S. 30-Year Treasury Bond.

Furthermore, it was found that a network’s forecasting accuracy depended on which and how many components with high variance are used, but does not necessarily depend on the number of training loops used. Percentage accuracy is highly correlated with capital gains/losses for SVMs but not for K-means clustering. The results also demonstrated that some networks were better at predicting weekly change in one direction than the other. K-means clustering can forecast up-weekly-changes better but experience maximum consecutive losses and SVMs can forecast down-weekly-changes better and experience maximum consecutive gain; this can be important to investors depending on which result is more important to him/her personally.

The investigation of the effectiveness of all three networks provides staggering results. Overall it can be said that the SVM networks are better suited because of higher percentage accuracy and its correlation to capital gains/losses; whereas for K-means clustering, percentage accuracy is correlated to neither ROI nor capital gains/losses. The SVMs may have performed better because it has been proven that non-linear data mapped to higher dimensions would eventually become linear.

## Future Work

This work is an explanatory tool for the application of pattern recognition systems as a financial forecasting solution. The general approach to this problem can be suited to other applications. There are several improvements that can be made to this specific system. The testing data set can be extended to include more than one year of data, since the long-term bond is usually held for longer than one year. As far as using RBF networks, the training data set can be reduced to use only the greatest influential factor; but this may prove to be difficult because any commodity in the stock/bond market is influenced by many factors. Also, seeing as how K-means clustering networks can vary the number of components used, the same should be done for SVMs and RBFs (instead of using all the feature components for each trial).

## REFERENCES

[1]. Afolabi, Mark O, Olude, Olatoyosi. “Predicting Stock Prices Using a Hybrid Kohonen Self Organizing Map (SOM),” Proceedings of the 40th Hawaii International Conference on System Sciences, 2007.

[2]. Anguita, D., Ridella, S., Sterpi, D. “A new method for multiclass support vector machines,” *Neural Networks,* Vol.1, 2004.

[3]. Boucher, Mark. The Hedge Fund Edge: Maximum Profit/Minimum Risk Global Trend Trading Strategies. New York City, New York: John Wiley & Sons, 1998.

[4]. Castellani, M. and Dos Santos, E. “Forecasting Long-Term Government Bond Yields: An Application of Statistical and AI Models,” 2006.

[5]. Chaveesuk, R., Srivaree-ratana, C. and Smith, A. E. ”Alternative Neural Network Approaches to Corporate Bond Rating,” *Journal of Engineering Valuation and Cost Analysis*, 1997.

[6]. Dutta, G., Jha, P., Laha, A.K., Mohan, N. “Artificial Neural Networks Models for Forecasting Stock Price Index in the Bombay Stock Exchange,” *Journal of Emerging Market Finance*, 2006.

[7]. Faber, Vance. “Clustering and the Continuous k-Means Algorithm,” Los Alamos Science. Vol 22, 1994, pp 138-144.

[8]. Falinoiuss, Pegah. “Stock Trend Prediction Using News Articles: A Text Mining Approach,” thesis, Luleå University of Technology, 2007.

[9]. Fan, Alan and Palaniswami, Marimuthu. “Stock Selection using Support Vector Machines,” IEEE 2001.

[10]. Gibaldi, Joseph. MLA Handbook for Writers of Research Papers. New York City, New York: The Modern Language Association of America, 2003.

[11]. He, X. and Lapedes, A. “Successive Approximation Radial Basis Function Networks for Nonlinear Modeling and Prediction,” *International Joint Conference on Neural Networks*, 1993.

[12]. Lee, Jae Won. “Stock Price Prediction Using Reinforcement Learning,” ISIE, 2001.

[13]. Lee, Raymond S. T. “iJade Stock Advisor: An Intelligent Agent Based Stock Prediction System Using Hybrid RBF Recurrent Networks,” *IEEE Transactions on Systems, Man, and Cybernetics. Part A – Systems and Humans*, Vol. 34, No. 3, May 2004.

[14]. McCluskey, Peter. “Feedforward and Recurrent Neural Networks and Genetic Programs for Stock Market and Time Series Forecasting,” thesis, Brown University, 1993.

[15]. Pavlidis, N.G., Plagianakos, V.P., Tasoulis, D.K., Vrahatis, M.N. “Financial Forecasting through Unsupervised Clustering and Neural Networks,” 2006.

[16]. Schalkoff, Robert. Pattern Recognition: Statistical, Structural, and Neural Approaches. New York City, New York: John Wiley & Sons, Inc., 1992.

[17]. Shlens, Jonathan. “A Tutorial on Principal Component Analysis,” Systems Neurobiology Laboratory, 2005.

[18]. Trafalis, T. and Ince, H. “Support Vector Machine for Regression and Applications to Financial Forecasting,” Volume 6, IEEE Computer Society, 2000, pp.348-353.

[19]. Van Bunningen, Arthur. “Augmented Trading,” thesis, University of Twente, 2004.

[20]. Wang, Li, ed. Support Vector Machines: Theory and Applications. Berlin: Springer-Verlag, 2005.

[21]. Ye, Qiang, Liang, Bing and Li, Yijun. “Amnestic Neural Network for Classification: Application on Stock Trend Prediction,” IEEE 2005.

[22]. www.supportvector.net

[23]. http://federalreserve.gov/releases

[24]. http://finance.yahoo.com

## BIOGRAPHICAL SKETCH

Nicole Powell was born in Chicago, Illinois in the year of 1985. She finished her high school education at Morgan Park High School as an International Baccalaureate student. She received her undergraduate degree from Florida Agricultural and Mechanical University in April of 2007. She majored in Computer Engineering.

Nicole was admitted to the Master’s Degree program in the Department of Electrical and Computer Engineering in Fall 2007. She specializes in the field of pattern recognition and data mining.