Journal of Advance Computational Research
Vol. 1 No. 2 (July-December, 2016)

Fuzzy Decision Tree Based Rule Extraction in Securities Analysis

W. Zheng, J. Cheng and M. Zargham
Department of Computer Science, Southern Illinois University Carbondale
Mailcode 4511, Carbondale, IL 62901
E-mail: chengjie_2000_2000@yahoo.com, mehdi@cs.siu.edu

Abstract: *While larger and larger pools of stock market data are available for investors, it is crucial for them to achieve the knowledge hidden behind and make the correct selections. The huge data amount, the variable data characteristic, and the noisy environment make this goal a great challenge. Using the model of fuzzy decision tree based rules extraction, a new set of fuzzy rules to select stocks with high returns is derived in this study. They are easy to comprehend and are proved to be efficient for picking these stocks. In performance evaluation, a portfolio is built for each year from year 1998 to year 2003 based on these rules. On average, the portfolio receives an annual return of 19.1%, much better than the average annual return, 5.8%, of the S&P 500 index.*

1. INTRODUCTION

In the stock market, larger and larger pools of historical data are available while useful information and knowledge is hidden behind. Securities analysis, which helps people predict the market's future behavior, has gained more and more attention in recent years. Several approaches have been proposed in this field. This section presents a brief review of these methods. Most approaches belong to three main categories: statistical models, neural networks, and rules induction.

Majority of statistical models applied to the stock market are time series. For example, the daily closing price of a share can be viewed as a time series. One aspect of time series analysis is trend analysis. A common method adopted to determine the trend of time series is moving average. The autoregressive integrated moving average (ARIMA) model proposed by Box, Jenkins, and Reinsel [1] is the most popular model of time series so far. Similarity Search is another kind of time series analysis. A similarity search looks for data sequences that differ only slightly from the given query pattern. Efficient similarity search in sequence database was studied by Agrawal, Faloutsos, and Swami [2]. This work was generalized in [3] to allow subsequence matching. There are several other research works on the search for similar sequences with respect to a variety of measures [4, 5, 6, 7]. Gavrilov, Anguelov, Indyk, and Motwani [8] studied various similarity measures for clustering similar stocks and revealed that appropriate normalizations of stock data resulted in better clustering performance.

Besides the disadvantage of time series model that it needs lots of data preprocessing such as segmentation and normalization, this model meets more problems when applied in stock markets. Firstly, the time series in stock markets, such as the closing price time series, behave much like a random-walk process. This is supported by the efficient market hypothesis (EMH). EMH suggests that securities price changes can only be explained by the arrival of new information, which, by definition, cannot be forecasted [9]. Furthermore, there are periods of "turbulence" in the stock market, in which the level of noise is greatly increased. The prediction model of time series, which works in the manner of the "moving window", will have its performance greatly affected when meeting or leaving this kind of turbulent period. The model of time series is very reliable and efficient in finding data sequences [3], however, it is generally believed that this model cannot provide very high prediction accuracy in stock markets.

Other than using the traditional statistical models including time series, many researchers have tried to apply neural network to securities analysis [10, 11, 12] and claimed that better prediction performance is gained [13]. Advantages of neural networks also include their high tolerance to noisy data as well as their ability to classify patterns on which they have not been trained [14]. However, this model has several aspects of limitations: First, since network design is a trial-and-error process and time-consuming, the best network architecture for a specific problem, such as stock selection, is difficult to find. Second, usually neural networks themselves need a long training time. Third, the acquired knowledge for a

neural network is contained in the network structure and the link weights makes it difficult to interpret or verify such knowledge. Available domain expert knowledge is also rather difficult to be incorporated into the system. To overcome this limitation, Lu, Setiono, and Liu [15] proposed the approach of NeuroRule to extract rules from neural networks. The generated rules, presented in the form of IF-THEN, are easy for humans to understand and manipulate. And with their experiment on simple classification functions, they claimed that their rules preserve the classification accuracy of the neural network. Nevertheless a complex rules extraction algorithm is needed in this approach and the whole model still suffers from the problem of difficult network design and long training time.

Rules are proved to be an appropriate tool for knowledge representation due to their high comprehensibility. There is a long tradition of attempting to use a set of rules to do stock selection, such as Peter Lynch investment rules, Warren Buffet investment rules, and Benjamin Graham investment rules. Among them, Graham's rules [16] are best known. These rules were developed by Graham to identify undervalued and miss-priced stocks and had their best performance before the year of 1976.

The present stock market, however, has been greatly developed and is quite different from the stock market back in the 1970s. Due to the weakness of stock prediction theories and the large amount of historical data, an efficient model is needed in order to develop new rules. Ren and Zargham [17] developed a new set of stock prediction rules based on the decision tree C4.5. Different from the situation in the model of neural networks, it is easy and quite straightforward to generate IF-THEN rules based on decision trees. Tracing a path from the root of the decision tree to a leaf node results in a rule, where the "IF" part consists of the conditions met along the path and the "THEN" part uses certain properties of the leaf node.

There are a series of algorithms for decision trees proposed, such as CART [18], ID3 [19], and C4.5 [20]. A variety of research works have been done to improve the performance of decision trees and justify their effectiveness [21, 22, 23]. However, traditional decision trees like CART [18], ID3 [19], and C4.5 [20] are not good at dealing with continuous attributes, whereas such attributes are very common in the stock market. The splitting for each continuous attribute in CART [18] is based on dynamically computed splitting points. In order to get the most appropriate splitting for one attribute, the algorithm needs to enumerate all possible splitting points for this attribute. ID3 [19] assumes that it works in the discrete domain so that an a-priori partitioning is needed for each continuous attribute. Its variant, C4.5 [20], calculates the thresholds for continuous attributes dynamically. Like CART [18], it also needs to exhaustively search for the best splitting point for each continuous attribute, which is not an efficient way. The model proposed in [17] also suffers from this problem and needs lots of raw data preprocessing. Fuzzy decision trees solves this problem satisfactorily. This method combines the symbolic decision trees with approximate reasoning provided by fuzzy representation and exploits the advantages of both [24]. Fuzzy decision trees inherit the high knowledge comprehensibility from traditional decision trees. At the same time, fuzzy representation grants fuzzy decision trees the ability to deal with inexact information or continuous values. In [25], a revised version of the ID3 algorithm, Fuzzy ID3, is proposed. To deal with continuous variables without predefined fuzzy terms, Janikow [26] presented the fuzzy partitioning mechanism in the fuzzy decision tree FID3.1. Comparative experiments proved that FID3.1 produce smaller tree size as well as better predictive accuracy than C4.5.

Accommodating to the fact that most data in the stock market are of continuous values, this research revise the traditional C4.5 decision tree to a fuzzy decision tree and applies it to the stock market data. Based on the generated decision tree, a mechanism is designed to derive a set of fuzzy classification rule for picking stocks with high returns. Data representing the fundamental variables in the stock market are used in the procedure. In the next section, fuzzy decision tree based on C4.5 [20] is explained in detail. Then the model of fuzzy decision tree based rules extraction is described in Section 3. Section 4 gives the overall summary, conclusion, and recommended future study.

2. FUZZY DECISION TREE

Different from traditional decision trees, the fuzzy decision tree splits based on linguistic variables. A linguistic variable is a variable whose values are words or sentences in the language. For example, "Height" is a linguistic variable when its values are "short", "middle" or "tall". These values of a linguistic variable are called fuzzy terms. Each fuzzy term corresponds to a fuzzy set and is characterized by its membership function.

In the fuzzy decision tree, when one internal node is split by a linguistic variable, the expanded branches of this node corresponds to the fuzzy terms. For example, let's look at a simple fuzzy decision tree below, which is used to decide if we can golf outside:

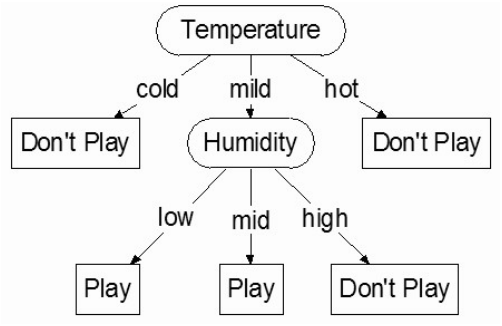


Figure 1: A Simple Fuzzy Decision Tree

The linguistic variable used in the root of the fuzzy decision tree is "Temperature". Its fuzzy terms are "cold", "mild" and "hot". One child of the root node can be divided by the linguistic variable "Humidity" further; corresponding fuzzy terms for this variable are "low", "mid" and "high".

In traditional decision trees such as C4.5 [20], when a training sample arrives at an internal node, it will go to only one of the output branches according to its value of the splitting attribute at that node. Contradictorily, in the case of fuzzy decision trees, a sample could fall into different branches with different membership values, which results in that a sample belongs to one node with a weight. Such a weight could be calculated along the path from the root to this node. A general framework for constructing a fuzzy decision tree has been proposed in [24]. Based on that framework and accommodating to our specific applications, an algorithm to construct a fuzzy decision tree is described as following:

(1) Generate the root node. Each sample belongs to the root node with the initial weight equal to one.

(2) A node N will be considered as a leaf and stop splitting when one of following criteria is met:
   (a) The samples in this node are of the same class.
   (b) The total weight of all samples in this node is smaller than a minimum size requirement. (In this study, this minimum size is set to four.)
   (c) There are no more attributes for splitting.

In our implementation, if one leaf node contains more than one class, we use the majority rule to decide the class of this leaf node instead of recording all classes. The class with the largest weight is assigned to this leaf node. The total weight of other classes is regarded as the error in this leaf node. Since we also use fuzzy variables for the classes in this research, when we calculate the weight of one class, we should consider both the weights of samples in this node and the membership values of samples for the class. Firstly, for each sample, multiply the weight of this sample with the membership value of this sample for the class. Secondly, get the sum of these products. This sum is considered as the weight of the class.

(3) For a node N which is needed to be expanded, select the splitting attribute using the criteria of information gain criteria:
   (a) For each linguistic variable V_i which has not been used along the path from the root to this node, evaluate the information gain ratio $GR^N(V_i)$:

$$GR^N(V_i) = (1 - UnknFrac^N(V_i)) * \frac{I^N - I^N(V_i)}{Split^N(V_i)}$$

   Here, I^N is the information content for this node, $I^N(V_i)$ is the weighted sum of information contents of all children nodes of node N, $Split^N(V_i)$ is the splitting information in node N with respect to the linguistic variable V_i and $UnknFrac^N(V_i)$ is the unknown fraction of samples in this node concerning V_i. The calculations for these values are similar to those in C4.5[20]; the difference is that during calculation, instead of using the count of samples, we use the weights of samples. The reason is that now a sample may belong to several nodes instead of fully belong to only one single node.
   (b) Compare all information gain ratios among these possible linguistic variables. Select the linguistic variable V_i such that the information gain ratio $GR^N(V_i)$ is maximal and split the node N using V_i. And then decide the new weight for each sample for all children nodes.

(4) Repeat steps 2 and 3 until no node can be split. In addition, after splitting, we set the error of one node N as the total error of all its children nodes. If this total error is larger than the error of N when it has not been split, we should cancel the splitting for N.

Based on the recursive algorithm described above, we construct our fuzzy decision tree using the data from the stock market in the next section. A new mechanism for rule extraction is also going to be presented.

3. FUZZY DECISION TREE BASED RULE EXTRACTION

This section illustrates how to apply the theory of fuzzy decision tree in securities analysis and how to generate fuzzy rules based on the constructed fuzzy decision tree. These rules are used to select stocks with high annual returns. The whole section consists of three parts: (1) Data Collection and Fuzzy Terms Design (2) Fuzzy Decision Tree Construction and Rules Extraction (3) Performance of Generated Rules.

3.1. Data Collection and Fuzzy Terms Design

In this research, a training data set extracted from Standard and Poor's COMPUSTAT database is used to construct the fuzzy decision tree. One company's data is treated as one sample. All 9820 active U. S. companies are included in the raw training set. The time period for the training set ranges from year 1990 to year 1997.

Each sample (company) has five attributes and one class label. The attributes used are five fundamental variables including Price to Earning Ratio Monthly (pem), Price to Book Ratio (mkbk), Current Ratio (CR), % Earning Growth (EG), and Earning Stability (ES). The class for each sample is its Annual Return (return). All the attributes and the class are in the format of numerical values in the raw data set. Not all the samples have values. The unknown values in each sample is marked with "?".

In this study, Annual Return (return) is used as the class attribute for each sample. Annual Return of one year is defined as the month-end close price in December of that year minus the month-end close price in January of the same year and then divided by the month-end close price in January. We suppose all the samples we used in this study should have Annual Return values. For those samples that can't have this value calculated due to lack of data in the database, we remove them from the pool of samples.

Our goal for this research is to pick the companies with good annual return; a return below zero is considered poor and a return above thirty percent is considered good. So three fuzzy terms are designed for Annual Return: "poor", "mid", and "good". The respective membership functions for Annual Return are plotted as below:

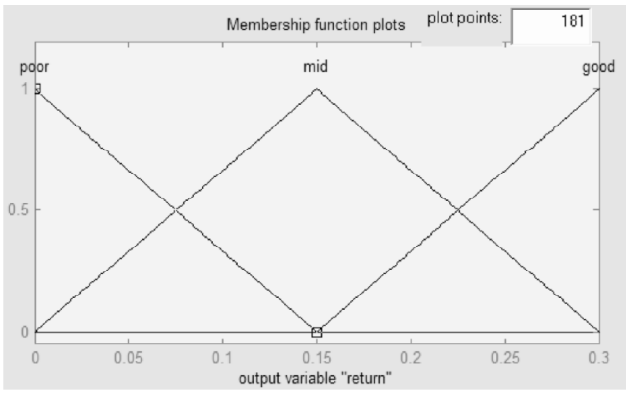
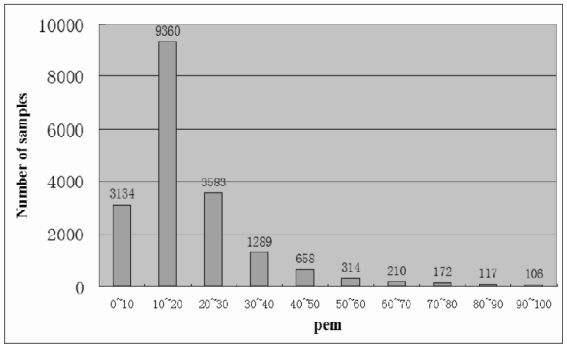
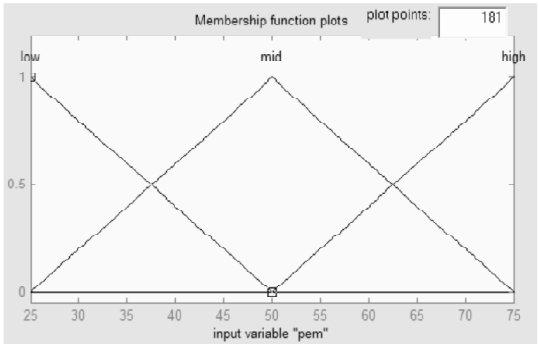
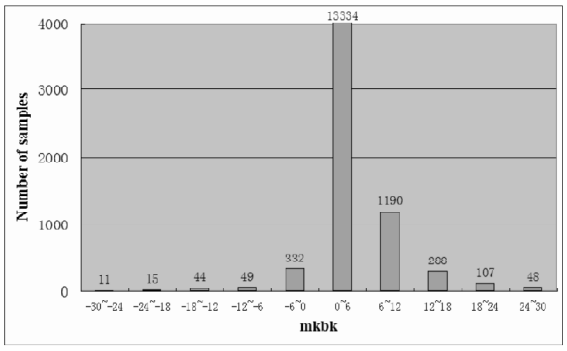
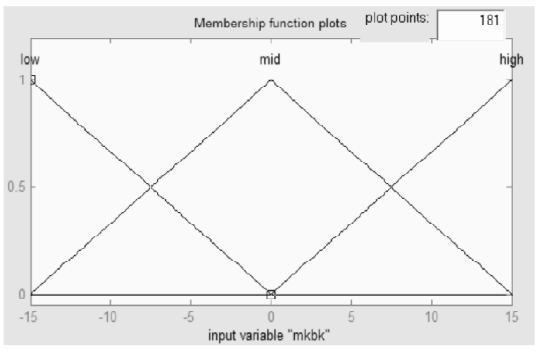
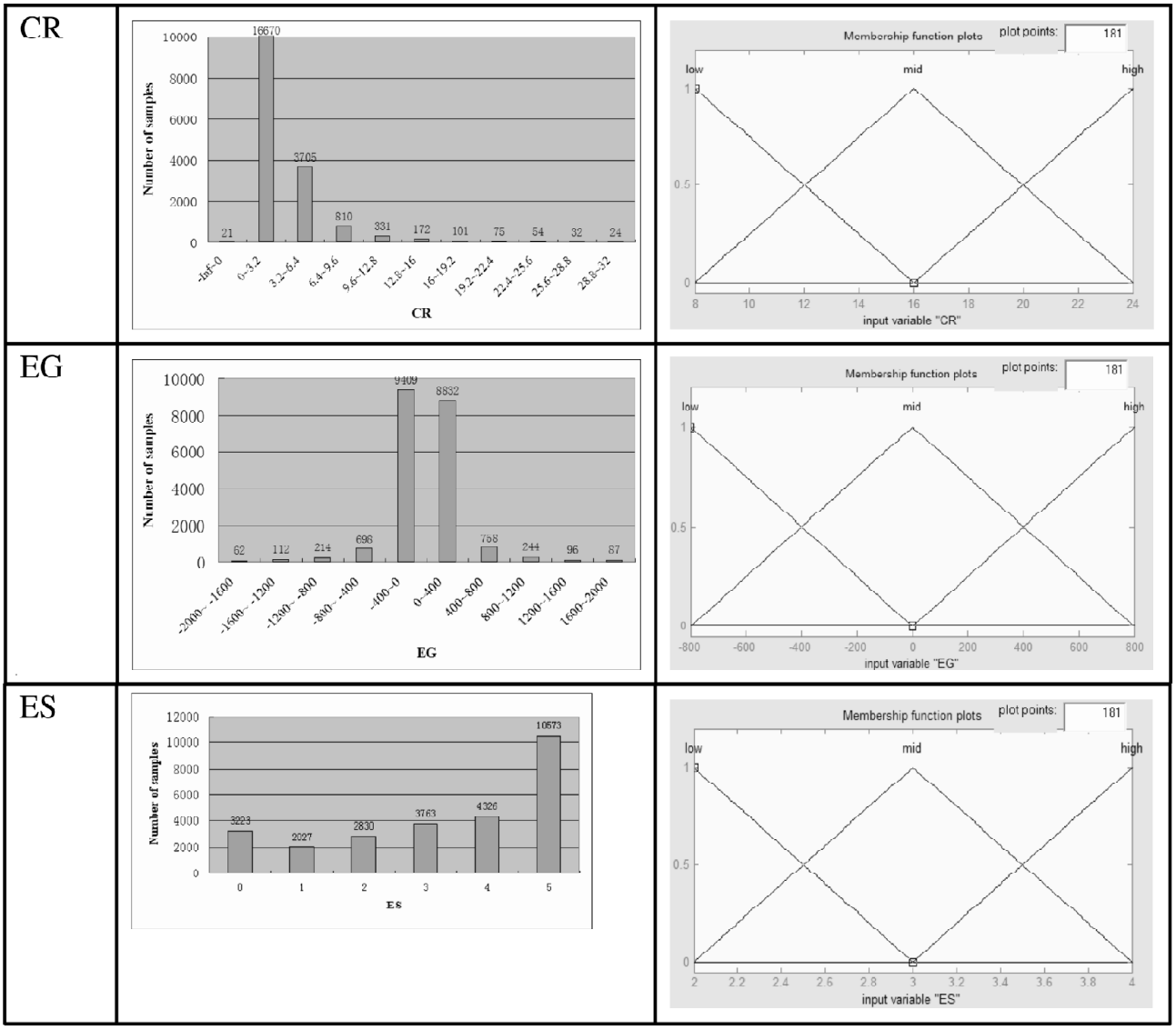


Figure 2: Membership Functions for Annual Return

In order to design the fuzzy terms for the five attributes reasonably, we first plot the distribution of these attributes (using the samples from year 1990 to 1997), and then based on the intervals that the majority of samples fall into, we create the fuzzy terms for these attributes respectively. The distributions as well as designed fuzzy terms for these five attributes are presented in the following table:

Att.	Distribution	Fuzzy Terms Plots
pem		
mkbk		



Besides the filtering we have already done at the beginning to remove samples (companies) with no Annual Return values, to assure the performance of our research, we need to further filter out some unqualified samples for our needs.

As we have already known, each sample has five attributes: pem, mkbk, CR, EG, ES. If more than half of the attribute values (i.e., larger than two) of one sample are unknown, we considered this sample an unqualified sample and removed it from our samples for constructing the fuzzy decision tree.

After the data filtering, we get 25320 samples from the raw data between year 1990 and year 1997. Parts of them are shown in the table below (The question marks mean unknown values). Based on these samples, along with the defined fuzzy terms for each attribute, the fuzzy decision tree is constructed.

Table 1
Parts of Input Samples

Price to Earning Ratio Monthly	*Price to Book Ratio*	*Current Ratio*	*% Earning Growth*	*Earning Stability*	*Annual Return*
2.338	0.729	6.219	100	3	2.0961
22.234	2.945	1.879	68.276	5	0.6191
15.337	2.434	0.952	1.805	5	0.1382
27.187	2.279	?	?	4	0.1328
6.293	2.67	1.724	?	3	0.1154
22.708	2.88	1.774	-16.667	5	0.0092
29.843	2.267	2.848	-66.387	4	-0.0103
22.222	3.449	1.649	-136.893	2	-0.0177
9.572	2.941	3.053	-381.013	1	-0.6516

3.2. Fuzzy Decision Tree Construction and Rules Extraction

The fuzzy decision tree construction algorithm is implemented based on the traditional C4.5 decision tree codes [20]. The main modifications include fuzzy attribute support, fuzzy splitting support, and fuzzy class support. The system takes two inputs: the file of attribute values and the file of fuzzy terms definitions. The output of the system is the desired fuzzy decision tree. Part of the output tree for our training data is given below:

```
Read 25320 cases (5 attributes) from Data90to97p.data

Decision Tree:

ES = low:
|   mkbk = low: Poor (2631.0/0.086)
|   mkbk = mid: Poor (6602.0/0.756)
|   mkbk = high:
|   |   CR = low: Poor (6285.0/0.256)
|   |   CR = high: Poor (856.0/0.070)
|   |   CR = mid:
|   |   |   EG = low: Good (938.0/0.007)
|   |   |   EG = mid: Poor (1075.0/0.054)
|   |   |   EG = high: Poor (726.0/0.013)
ES = mid:
|   mkbk = low: Poor (1322.0/0.021)
|   mkbk = mid:
|   |   pem = low:
|   |   |   CR = mid: Poor (603.0/0.044)
|   |   |   CR = high: Poor (542.0/0.029)
|   |   |   CR = low:
|   |   |   |   EG = low: Poor (2917.0/0.228)
|   |   |   |   EG = mid: Poor (3002.0/0.524)
```

Figure 3: Generated Fuzzy Decision Tree

The constructed tree is similar to the decision trees generated by traditional C4.5 [20]. The difference is that the internal nodes here are split by fuzzy restrictions instead of crisp conditions and the leaves here are of fuzzy classes.

In Figure 3, there are two numbers for each leaf N (separated by the symbol /): the first one is NUM^N, the number of the samples in this leaf; the second one is RT^N, the ratio of the total weight of all samples in N to the number of samples in N. These two values are used to select the appropriate leaves in order to generate corresponding rules. The value of NUM^N illustrates how many samples will arrive at this leaf, i.e., how many samples support this leaf. As we know, in the fuzzy decision tree, a sample could belong to several leaves at the same time with different weights instead of fully belonging to one leaf. The value of RT^N shows how much on average a sample belongs to this leaf N. Our selection mechanism uses these two values as the selection criteria and works based on the steps given in Figure 4.

Here the threshold A is set as twenty percent of total input samples, which is 5064; the threshold B is set as ten percent. In total six leaves are picked. As in traditional decision trees, the path to each leaf could be transformed into an "IF-THEN" rule by tracing all the test outcomes along the path. Six generated rules are shown below:

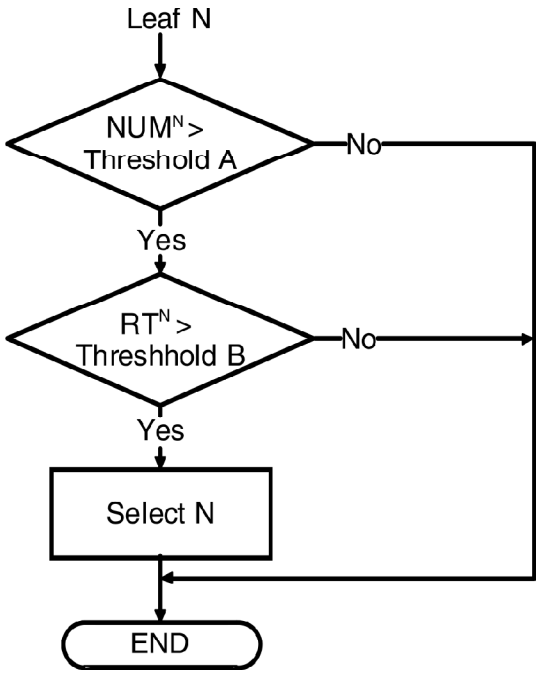


Figure 4: Flowchart for Rules Extraction

(1) IF (mkbk is mid) and (ES is low) THEN return is poor.

(2) IF (mkbk is high) and (CR is low) and (ES is low) THEN return is poor.

(3) IF (pem is low) and (mkbk is mid) and (EG is low) and (ES is high) THEN return is poor.

(4) IF (pem is low) and (mkbk is mid) and (EG is mid) and (ES is high) THEN return is poor.

(5) IF (pem is low) and (mkbk is high) and (CR is low) and (EG is mid) and (ES is high) THEN return is good.

(6) IF (pem is low) and (mkbk is mid) and (CR is low) and (EG is high) and (ES is high) THEN return is good.

3.3. Rules Performance Analysis

To evaluate the performance of generated rules, a fuzzy system utilizing these rules is built in MATLAB 6.5. The main components of the system are described as follows: (1) The input variables of this system are Price to Earning Ratio Monthly (pem), Price to Book Ratio (mkbk), Current Ratio (CR), % Earning Growth (EG), Earning Stability (ES). (2) The output variable is Annual Return (return). The definitions for membership functions of these variables are the same as described before. (3) For the six rules, the "and" method is interpreted as "min"; the "or" method is interpreted as "max"; the implication uses "min"; the aggregation among results of rules uses "max"; and the method of centroid is used for

defuzzification. The overview of the whole system is shown in Figure 5.

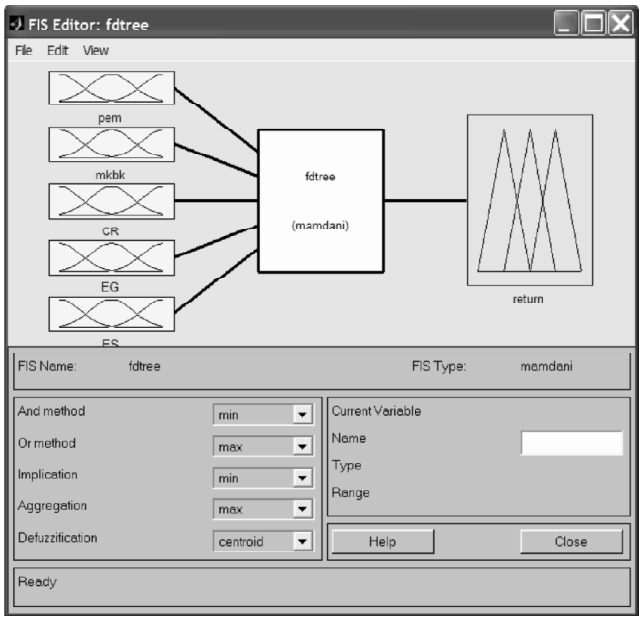


Figure 5: Overview of the Fuzzy System

Suppose we have a sample whose pem is 25, mkbk is 4.388, CR is 8, EG is 760, and ES is 4. The roadmap of the whole fuzzy inference process for this sample is shown in the following figure, where each row stands for a rule, and each column is a variable. The last column shows the THEN part of each rule; previous columns show the IF part of each rule. The seventh plot in the last column shows the final decision of the system, which is "return = 0.231".

Test data ranging from year 1997 to year 2002 are used in evaluation to create portfolios from year 1998 to

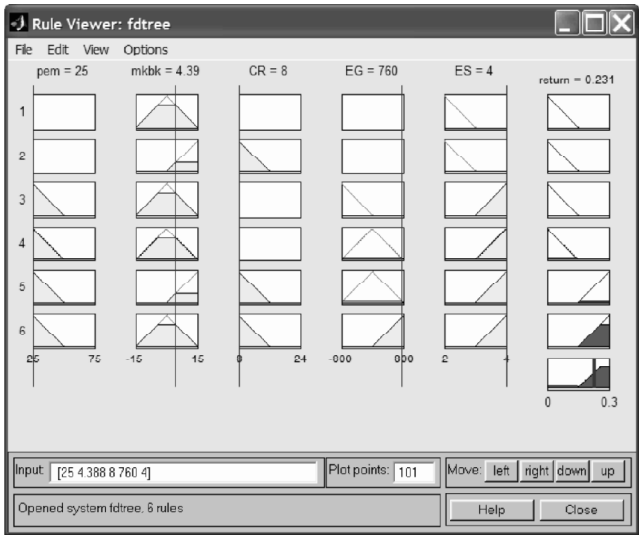


Figure 6: Fuzzy Inference Process for One Sample

year 2003 respectively. These input data are also from Standard and Poor's COMPUSTAT database and belong to the set of all active U. S. companies. Extreme items with EG > 1000 or EG < -1000 are filtered out in the beginning of the process. For each year, a portfolio is built according to the output of our fuzzy inference system. Any samples with a system output not smaller than twenty percent will be put in the portfolio.

The following table shows the performance of constructed portfolios. For each year from year 1998 to year 2003, the average annual return among companies of our portfolio is listed, together with the number of stocks selected into the portfolio. For example, the portfolio for year 1998, created by using data of year 1997, consists of 61 companies. For each company, we calculate the annual return by comparing the month-end close prices between January and December 1998. After summing up the 1998 annual returns for these companies and dividing it by 61, we get the average annual return for this portfolio: 0.106 (10.6%). For comparison, the annual returns of S&P 500 Index are listed in the last column of the table. S&P 500 is an index of 500 stocks chosen for their market value, liquidity and industry group representation. It is usually considered a benchmark for the overall U. S. stock market performance and does better in Annual Returns than most mutual funds.

Table 2
Performance of Portfolios

	Number of Selected Stocks	*Average Annual Return*	*Annual returns of S&P 500 Index*
Portfolio 1998	61	0.106	0.285
Portfolio 1999	61	0.202	0.210
Portfolio 2000	69	0.056	- 0.091
Portfolio 2001	62	0.179	- 0.119
Portfolio 2002	47	-0.020	- 0.221
Portfolio 2003	49	0.621	0.286
Average	58.2	0.191	0.058

As we could observe in Table 2, in total six portfolios are built, with an average size of 58.2. On average, the portfolio receives an annual return of 19.1%, much better than the average annual return, 5.8%, of the S&P 500 index.

4. CONCLUSIONS

In order to derive a new set of rules to select stocks with high returns, the method of fuzzy decision tree based rules extraction is adopted in this thesis. The whole system is implemented in four phases: data collection and fuzzy

terms design, fuzzy decision tree construction, rules extraction, and performance analysis.

The performance analysis for these rules shows that fuzzy decision tree based rules extraction is an efficient model to extract rules for stock selection. As presented in the last section, in total six fuzzy classification rules have been extracted in this study using training data ranging from year 1990 to year 1997. A fuzzy system is built based on these six fuzzy rules. According to the output of this fuzzy system, a portfolio is created for each year from year 1998 to year 2003. The portfolio receives an average annual return of 19.1%, much better than the average annual return, 5.8%, of the S&P 500 index.

Our model of fuzzy decision tree based rules extraction gives a way to process the stock data of continuous characteristic and then make stock selections. It can be concluded that such a model has the capability to reveal certain hidden patterns in how to select the stocks with high returns using the fundamental data of the stock market.

In the future, more fundamental variables available in the stock market can be introduced into this model. At the same time, if the resulted rules become too trivial, rules pruning should be considered. Another possible improvement might be the automatic definitions for the fuzzy terms as well as their membership functions by incorporating the fuzzy partitioning algorithms. Finally, other types of traditional decision trees could also combined with approximate reasoning to build another kind of fuzzy decision tree. Comparative study can be done to explore if there is a most appropriate one for the application in securities analysis.

REFERENCES

[1] G. Box, G. M. Jenkins, and G. Reinsel, "Time Series Analysis: Forecasting & Control", 3rd Edition, Prentice-Hall, 1994.

[2] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient Similarity Search in Sequence Databases", In Proc. of the 4th International Conference of Foundations of Data Organization and Algorithms (FODO), Chicago, Oct. 1993.

[3] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast Subsequence Matching in Time-Series Databases", In Proc. of ACM SIGMOD Conference on Management of Data, Mineapolis, MN, May, 1994.

[4] B. Bollobas, G. Das, D. Gunopulos, and H. Mannila, "Time-Series Similarity Problems and Well-Separated Geometric Sets", Proc. of Symposium on Computational Geometry 1997, 454-456.

[5] G. Das, D. Gunopulos, and H. Mannila, "Finding Similar Time Series", Proc. of KDD 1997, 88-100.

[6] D. Rafiei and A. Mendelzon, "Similarity-Based Queries for Time Series Data", SIGMOD'97, Tucson, Arizona, May 1997.

[7] B. K. Yi, H. V. Jagadish, and C. Faloutsos, "Efficient Retrieval of Similar Time Sequences Under Time Warping", ICDE'98, Orlando, FL, Feb. 1998.

[8] M. Gavrilov, D. Anguelov, P. Indyk, and R. Motwani, "Mining the Stock Market: Which Measure is Best?", Proc. of the Sixth Int. Conf. Knowledge Discovery and Data Mining (KDD), 487-496, 2000.

[9] A. Lendasse, E. de Bodt, V. Wertz, M. Verleysen, "Non-linear Financial Time Series Forecasting Application to the Bel 20 Stock Market Index", *European Journal of Economic and Social Systems*, **14**, 81-91, 2000.

[10] Kryzanowski, Lawrence, M. Galler, and D. W. Wright. "Using Artificial Neural Networks to Pick Stocks", *Financial Analysts Journal*, **49**(4), 1993, 21-27.

[11] Grundnitski, Gary and L. Osburn. "Forecasting S&P and Gold Futures Prices: An Application of Neural Networks", *Journal of Futures Markets*, 1993, **13**(6), 631-643.

[12] A. N. Refenes, A. Zapranis, G. Francis, "Stock Performance Modeling Using Neural Networks: A Comparative Study with Regression Models", *Neural Networks*, **7**(2), 1994, 375-388.

[13] M. Adya and F. Collopy, "How Effective are Neural Networks at Forecasting and Prediction? A Review and Evaluation", *Journal of Forecasting*, **17**(5-6), 1998, 481-495.

[14] J. Han, and M. Kamber, "Data Mining: Concepts and Techniques", Morgan Kaufmann, 2001.

[15] H. J. Lu, R. Setiono, and H. Liu, "NeuroRule: A Connectionist Approach to Data Mining", In Proc. of the 21st VLDB Conference, Zurich, Switzerland, 1995, 478-489.

[16] P. Blustein, "Ben Graham's Last Will and Testament", Forbes, August 1, 1977, 43-45.

[17] N. Ren, "Rule Extraction for Security Analysis Based on Decision Tree Classification Model", Master's Thesis, Southern Illinois University Carbondale, Major Professor: M. Zargham, May 2004.

[18] L. Breiman, J.H. Friedman, R. A. Olsen, and C. J. Stone, "Classification and Regression Tree", Belmont, CA: Wadsworth, 1984.

[19] J. R. Quinlan, "Induction of Decision Trees", Machine Learning, **1**, 1986, 81-106.

[20] J. R. Quinlan, "C4.5: Programs for Machine Learning", San Mateo, CA: Morgan Kaufmann, 1993.

[21] G. H. John, "Robust Decision Trees: Removing Outliers from Databases", in Proc. of First International Conference on Knowledge Discovery and Data Mining, Montreal, Canada, 1995, 174-179.

[22] M. Mehta, J. Rissanen, R. Agrawal, "MDL-based Decision Tree Pruning", in Proc. of First International Conference on Knowledge Discovery and Data Mining, Montreal, Canada, August 1995, 216-221.

[23] S. Murthy, S. Salzberg, "Decision Tree Induction: How Effective is the Greedy Heuristic?", in Proc. of First International Conference on Knowledge Discovery and Data Mining, Montreal, Canada, 1995, 222-227.

[24] C. Z. Janikow, "Fuzzy Decision Trees: Issues and Methods", *IEEE Transactions on Systems, Man and Cybernetics*, Part B, **28**(1), 1998, 1-14.

[25] M. Umano, H. Okamoto, I. Hatono, H. Tamura, F. Kawachi, S. Umedzu, and J. Kninoshita, "Fuzzy Decision Trees by Fuzzy ID3 Algorithm and Its Application to Diagnosis Systems", *Proc. of the Third IEEE Conference on Fuzzy Systems*, 1994, **3**, 2113-2118.

[26] C. Z. Janikow and M. Fajfer, "Fuzzy Partitioning with FID3.1" in Proc. of the 18th International Conference of the North American Fuzzy Information Processing Society, 1999, 467–471.