Financial Computational Intelligence

Chiu-Che Tseng, Yu-Chieh Lin
Department of Computer Science and Information Systems
Texas A&M University – Commerce, TX 75429, U.S.A.

Abstract:

Artificial intelligence decision support system is always a popular topic in providing the human with an optimized decision recommendation when operating under uncertainty in complex environments. The particular focus of our discussion is to compare different methods of artificial intelligence decision support systems in the investment domain – the goal of investment decision-making is to select an optimal portfolio that satisfies the investor's objective, or, in other words, to maximize the investment returns under the constraints given by investors. In this study we apply several artificial intelligence systems like Influence Diagram (a special type of Bayesian network), Decision Tree and Neural Network to get experimental comparison analysis to help users to intelligently select the best portfolio.

1. Introduction

The investment domain, like many other domains, is a dynamically changing, stochastic and unpredictable environment. Take the stock market as an example; there are more than two thousand stocks available for a portfolio manager or individual investor to select. This poses a problem of filtering all those available stocks to find the ones that are worth investment. There are also vast amounts of information available that will affect the market to some degree.

For these problems, artificial intelligence decision support systems are always the solutions. The decision support systems provide the investor with the best decision support under time constraints. For this purpose, we use the Influence Diagram, Decision Tree and Neural Network to advice users to build their own highly successful investment portfolios.

The structure of the paper is as follow. In section 2, we introduce some related works on the structures of the investment decisions for portfolio managements. In section 3 and 4, we describe the frameworks of the Influence Diagram, decision tree and neural networks. In section 5, we specify our experimental settings. In section 6, we show our experimental results and explanations. And in section 7, we conclude our paper.

2. Related Work

We explore several ways to reduce the complexity of the investment decision deliberation that might cause investors to lose money under urgent situations, and, at the same time, to provide the highest quality investment recommendations possible.

For portfolio management, Strong [20], Reilly and Norton [15] and Jones [10] brought several traditional portfolio management strategies. And Sycara, et al. [21] focused on using distributed agents to manage investment portfolios. Their system deployed a group of agents with different functionality and coordinated them under case-based situations. They modeled the user, task and situation as different cases, so their system activated the distributed agents for information gathering, filtering and processing based on the given case. Their approach mainly focused on portfolio monitoring issues and has no mechanism to deal with uncertainty. Our systems on the other hand react to the real-time market situation and gather the relevant information as needed. John, et al. [9] made extensive research on stock selections by applying induction rules into data mining applications. Other related research on portfolio selection problems has received considerable attention in both financial and statistics literature; see Cover [3]and Cover, et al, [4].

In the field of model refinement, there are several approaches. The value of modeling was first addressed by Watson and Brown [26] and Nickerson and Boyd [12]. Chang and Fung [2] considered the problem of dynamically refining and coarsening of the state variables in Bayesian networks. However, the value and cost of performing the operations were not addressed. Control of reasoning and rational decision making under resource constraints, using analyses of the expected value of computation and consideration of decisions on the use of alternative strategies and allocations of effort, has been explored by Horvitz [7] and Russell and Wefald [18] Poh and Horvitz [13] explored the concept of expected value of refinement and applied it to structural, conceptual

and quantitative refinements. Their work concentrated on providing a computational method for the criteria to perform the refinement. However, their work did not address the need of a guided algorithm to perform the node refinement throughout the network. In our previous work Tseng, et al, [25], we used guided methods to perform the conceptual refinement for our models. We see significant performance improvement of the models after applying the refinement algorithm. And Tseng [23,24] made extensive researches on applying influence diagram into portfolio selections and comparing our intelligence decision support system with other artificial intelligence systems, such as C5.0, and made conclusions that our system will handle better than C5.0 in the dynamic environment. Starzyk, et al, [19] also brought Self-Organizing Learning Array system into economical and financial applications by comparing our influence diagrams decision support system.

Besides there is few work on the comparison of different intelligence decision support systems. In our paper you will see how different systems work differently.

3. Influence Diagram

An influence diagram is a special type of Bayesian network (Figure 1), one that contains the decision node and the utility node to provide a decision recommendation from the model. Influence diagrams are directed acyclic graphs with three types of nodes—chance nodes, decision nodes and utility nodes. Chance nodes, usually shown as ovals, represent random variables in the environment. Decision nodes, usually shown as squares, represent the choices available to the decision-maker. Utility nodes, usually of diamond or flattened hexagon shape, represent the usefulness of the consequences of the decisions measured on a numerical utility scale. The arcs in the graph have different meanings based on their destinations. Dependency arcs are the arcs that point to utility or chance nodes representing probability or functional dependence. Informational arcs are the arcs that point to the decision nodes implying that the pointing nodes will be known to the decision-maker before the decision is made.

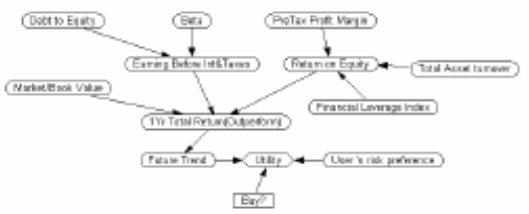

Figure 1. A simple influence diagram

When using an influence diagram for decision support problems, there are some fundamental characteristics of the influence diagram that one must take into consideration. These characteristics influence the data requirements and the choice of the appropriate influence method. The first characteristic is the granularity of the values for each node. This characteristic affects the memory requirement for storing the probabilities and the computational time required for updating the probabilities. The more values within each node, the larger the memory required and the longer it will take to propagate the probability update. The second characteristic is the integration of the user's preference into the utility node. This characteristic will affect the decision outcome of the model. Given different preferences among users, the model might return a different decision recommendation. Another issue of this characteristic is how to model the user's preference into a set of values for the utility node. Different fields of research have suggested different approaches for this problem. Some suggest learning from the user's behavior, some suggest obtaining data from a user survey, and some simply query the expert and assign subjective values.

The third characteristic to consider is the availability of the knowledge about the structure, probabilistic knowledge for the prior and the conditional probabilities. There are many variables in a specific problem domain and several concepts might exist in the problem domain that are observationally equivalent, which means they are not distinguishable even with infinite data. To find out which of those are relevant to the problem and the casual relationships among them present a challenge to the knowledge engineer. There has been much research and many tools devoted to the learning of the model structure from the data. [4] For the probability distribution for the node, there are two methods for obtaining the probabilities. First, the probability distributions can be based on frequency by obtaining the data from gathered statistics. The second method is to obtain the probability distributions through knowledge acquisition sessions from the domain experts, who convey their subjective beliefs. In both cases, the probabilities can be refined through a feedback mechanism. Finally, the size, topology and connectivity of the model should also be considered. Applying good knowledge engineering techniques [8] throughout the construction of the model will help keep the network manageable.

4. Decision Tree Algorithm

We chose to use decision trees because they provide a comprehensible representation of their classification decisions. Although techniques such as boosting [6, 17] or support vector machines might obtain slightly higher classification accuracy, they require more computation during classification and they further obscure the decision making process.

A decision tree is a tree structure where each internal node denotes a test on a feature, each branch indicates an outcome of the test, and the leaf nodes represent class labels. An example decision tree is shown in Figure 2. To classify

an observation, the *root* node tests the the value of feature *A*. If the outcome is greater than some value *x*, the observation is given a label of *Class 1*. If not, we descend the right subtree and test the value for feature *B*. Tests continue until a leaf node is reached. The label at the leaf node provides the class label for that observation.

We chose to use the C5.0 decision tree algorithm[14] a widely used and tested implementation. For details regarding the specifics of C5.0 the reader is referred to[14, 16]. Here we provide only the key aspects of the algorithm related to decision tree estimation, particularly as it pertains to feature selection. The most important element of the decision tree estimation algorithm is the method used to estimate splits at each internal node of the tree. To do this C5.0 uses a metric called the information gain ratio that measures the reduction in entropy in the data produced by a split. In this framework, the test at each node within a tree is selected based on splits of the training data that maximize the reduction in entropy of the descendant nodes. Using these criteria, the training data is recursively split such that the gain ratio is maximized at each node of the tree. This procedure continues until each leaf node contains only examples of a single class or no gain in information is given by further testing. The result is often a very large, complex tree that overfits the training data. If the training data contains errors, then overfitting the tree to the data in this manner can lead to poor performance on unseen data. Therefore, the tree must be pruned back to reduce classification errors when data outside of the training set are to be classified. To address this problem C5.0 uses confidence-based pruning[14].

When using the decision tree to classify unseen examples, C5.0 supplies both a class label and a confidence value for its prediction. The confidence value is a decimal number ranging from zero to one – one meaning the highest confidence – and it is given for each instance.

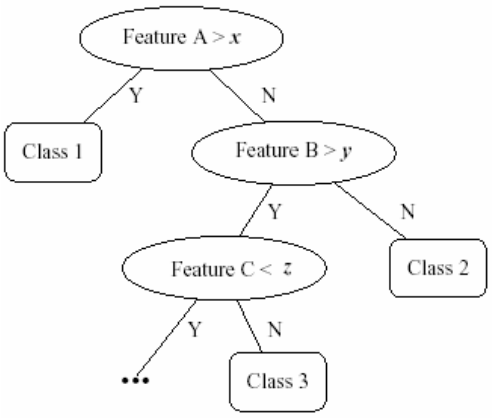

Figure 2. Decision tree abstraction showing how the values associated with certain features determine the class label. In this example, observations whose value for feature A is greater than X are assigned a class label of Class1. Other classifications are based on the values of features B and C.

```
d5 > 0.0719:
:...d2 > 0.0347:
:   :...d2 <= 0.0415:
:   :   :...d5 <= 0.0833: -1 (4)
:   :   :   d5 > 0.0833: 1 (4/1)
:   :   d2 > 0.0415:
:   :   :...d3 <= 0.0188: -1 (5/2)
:   :       d3 > 0.0188: 5 (4)
:   d2 <= 0.0347:
:   :...d1 > 0.0583: 5 (5/1)
:       d1 <= 0.0583:
:       :...d2 <= -0.031:
:           :...d4 <= -0.0197: 0 (4/1)
:           :   d4 > -0.0197:
:           :   :...d2 <= -0.0483: -5 (3)
:           :       d2 > -0.0483:
:           :       :...d3 <= 0.0104: -3 (3/1)
:           :           d3 > 0.0104: 0.5 (3)
```

Figure 3. Portion of a decision tree generated by C 5.0

5. Neural Network

A neural network is a powerful data modeling tool that is able to capture and represent complex input/output relationships. The motivation for the development of neural network technology stemmed from the desire to develop an artificial system that could perform "intelligent" tasks similar to those performed by the human brain. Neural networks resemble the human brain in the following two ways:

1. A neural network acquires knowledge through learning.
2. A neural network's knowledge is stored within inter-neuron connection strengths known as synaptic weights.

The true power and advantage of neural networks lies in their ability to represent both linear and non-linear relationships and in their ability to learn these relationships directly from the data being modeled.

The most common neural network model is the multilayer perceptron (MLP). This type of neural network is known as a supervised network because it requires a desired output in order to learn. The goal of this type of network is to create a model that correctly maps the input to the output using historical data so that the model can then be used to produce the output when the desired output is unknown. A graphical representation of an MLP is shown below in Figure 4.

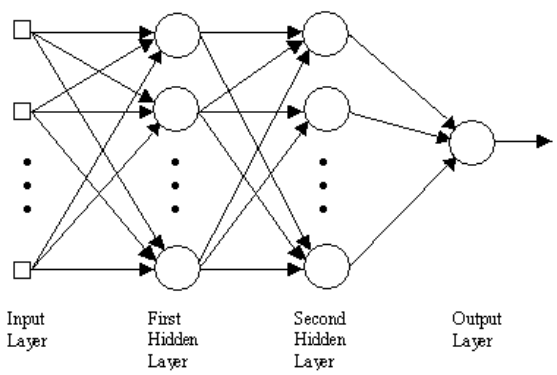

Figure 4. A MLP neural network model

In neural networks, an activation function is the function that describes the output behaviour of a neuron. The most common activation functions are sigmoid and Gaussian functions.

- Sigmoid function $f(\mu_i) = \dfrac{1}{1+e^{-\mu_i/\sigma}}$
- Gaussian function $f(\mu_i) = ce^{-\mu_i^2/\sigma^2}$

6. Experiment Settings

6.1 Influence Diagram Model

Our influence diagram model of the investment domain consists of a number of stocks for the investors to construct an investment portfolio. The goal is to maximize the profit from the investment portfolio. The experiments are written in C++ and built on top of the Netica Belief network package, running on a LINUX platform.

In the experiments we ran, we selected eight financial ratio data from the S&P 500 companies as the input factors to the system. The training data is collected from the Compustat database from the period of 1998 to 2002. To test the performance, we used the date from the year 2003 and let the system make the decision recommendation on which of the S&P 500 companies should be included in the investment portfolio.

6.2 Neural Network Model

Before a neural network can be trained with these data, they must be normalised, or called in pre-processing phase. The following rule has been used for normalisation derived from Angstenberger (1996):

1. Multiplication of the data by the factor: $\dfrac{0.8}{\max - \min}$;
2. Addition of the offset 0.1 – (factor * min);

In this way the data can be scaled to the range from 0.1 to 0.9. The trained data can also be transformed back to the original scale, due to the linearity of the formula.

The experiments of neural network model are built on top of the Brainmaker Professional network package, running on a WINDOWS platform. There are 2500 total training data sets, which are eight financial ratio data from the S&P 500 companies collected from the Compustat database from the period of 1998 to 2002. There are 500 total testing data sets, which are the same categories as the training data except they are from the period of 2003. We conducted extensive experiments and found that the one with the best result used the following parameters:

- Number of hidden layers: 1
- Input layer 1: Gaussian, 338 neurons
- Output layer: Sigmoid
- Learning rate: Exponential, 1.0

6.3 Decision Tree Model

Before training, the data will be normalized and change into the format which could be recognized by C 5.0. The following are the data used in the training set:

Data Variation: the data column/ the max data value of the same data column

6.3.1 Specifying the classes

C5's job is to find how to predict a case's class from the values of the other attributes. C5 does this by constructing a *classifier* that makes this prediction. As we will see, C5 can construct classifiers expressed as *decision trees.*

Because of that, we define 2 classes.
Class " 1 ": The one year total return of the company is greater than or equal to the average of the one year return of the S&P 500 companies.
Class " 0 ": The one year total return of the company is smaller than the average of the one year return of the S&P 500 companies.

6.3.2 Training Process

Decision tree learning follows a kind of top-down, divide-and-conquer learning process. The basic algorithm for decision tree learning can be described as follows:

1. Based on an information gain measure, select an attribute to place at the root of the tree and branch for each possible value of the tree. Thereby, the underlying case set is split up into subsets, one for each value of the considered attribute.

2. Recursively repeats this process for each branch, using only those cases that actually reach that branch.

3. If at any time all instances at a node have the same classification, stop developing that part of the tree.

7. Results

The experiment results are summarized in table 1 shown the 2003 performance of each artificial intelligence technique that are used to select stocks from S&P 500. In table 2 shown the 2003 performance of combining two or three artificial intelligence techniques that are used to select stocks from S&P 500. The following four subsections describe the detail result from each architecture and combine two or three architectures.

7.1 Influence Diagram Model

On the 2003 test data, the influence diagram portfolio obtains its maximum average annual return performance when the portfolio contains 108 companies out of the 500. The 108 companies produced an average one-year total return of 32.28%, while the average return of all the 500 companies was 41.35%.

7.2 Neural Network Model

Unlike our first model, this portfolio obtains its maximum one year total return performance when the portfolio contains 447 companies out of the 500 companies on the 2003 test data. This portfolio produces an average one year total return of 40.86%, which is lower than 41.35%, the average of the S&P500's return. The root mean square error is 0.5032, which is too high to get any reliable results.

It is true that the quality of the forecast is highly dependent on the network parameters. But after carefully analyzed our data, we found that there exist some cross-sectional problems. The 500 companies are chosen from different industries fields and each has its own economic cycles. For example, all the local "dot com" companies were in their recession cycles at the beginning of this century, while the Chinese companies were in their booming cycles, like SOHU and SINA. Besides, the "size effect" [15] is also a critical factor affecting results. The sizes of the companies under S&P500 index are quite different, which make their returns different between large-size companies and small-size companies. All these problems can not be easily distinguished in the general neural network structures.

7.3 Decision Tree Model

This portfolio obtains its maximum one year total return performance when the portfolio contains 165 companies out of the 500 companies on the 2003 test data. This portfolio produces an average one year total return of 46.38%. The result is higher than 41.35%, the average of the S&P500's return.

In our decision tree model, we have the most reliable result of the one year total return among all the intelligence systems. Nevertheless, the stock market is extremely sensitive to its environment, and many objects related to the stock market contribute their patterns to the stock price. Our goal is to extract patterns related to each object and build a model of the object from these patterns.

7.4 Self-Organizing Learning Array (SOLAR) system

Starzyk, et al [19] compared our work with their machine learning technique in (SOLAR) system: A new Self-Organizing Learning Array (SOLAR) system. SOLAR is capable of handling a wide variety of classification problems. It has a regular array structure with sparsely interconnected computing elements and local learning rules. Computing elements choose their connections and define their own functionality, while learning how to best extract information from their input data.

SOLAR was constructed without expertise in this field and has not been refined for this specific problem. Nine individual SOLAR networks were used in this case, each of which yielded 17% to 84% independently, which still shows higher volatility and stochasticity. However, its high average yield made itself worth further research on portfolio selections, though, in the Adult Income Classification case [19], Bayes algorithms outperformed SOLAR slightly.

Table 1. Result from each architecture

	Average one year total return
Influence Diagram	32.28%
Neural Network	40.86%
Decision Tree	46.38%
SOLAR	30.10%

7.5 Combining Prediction

We select those companies that are selected by two or three artificial intelligence techniques we use at the same time to get the average one year total return. By using decision tree and influence diagram, the portfolio obtains its maximum one year total return performance when the portfolio contains 22 companies out of the 500 companies on the 2003 test data. This portfolio produces an average one year total return of 28.78%. The result is lower than 41.35%, the average of the S&P500's return. By using influence diagram and neural network, the portfolio obtains its maximum one year total return performance when the portfolio contains 99 companies out of the 500 companies on the 2003 test data. This portfolio produces an average one year total return of 33.36%. The result is lower than 41.35%, the average of the S&P500's return. By using decision tree and neural network, the portfolio obtains its maximum one year total return performance when the portfolio contains 146 companies out of the 500 companies on the 2003 test data. This portfolio produces an average one year total return of 47.26%. The result is higher than

41.35%, the average of the S&P500's return. By using decision Tree, influence diagram and neural network, the portfolio obtains its maximum one year total return performance when the portfolio contains 19 companies out of the 500 companies on the 2003 test data. This portfolio produces an average one year total return of 29.07%. The result is lower than 41.35%, the average of the S&P500's return.

Table 2. Result from combining two or three architectures

	Average one year total return
Decision Tree & Influence Diagram	28.78%
Influence Diagram & Neural Network	33.36%
Decision Tree & Neural Network	47.26%
Decision Tree & Influence Diagram & Neural Network	29.07%

8. Conclusion

We conducted some performance analysis with our systems and compared ours with other intelligence system. Our decision support system uses the decision tree as the decision model; the structural information of the decision tree plays an important role on the performance of our system. We obtained the structural information from the domain expert and the information represents what the expert's opinion on the causal relationships among the nodes. From the experiment results, we can see that the decision tree system works better than many other artificial intelligence systems in a more general situation. This is due to the background information given by the domain expert when constructing the network.

Given the above analysis, we could conclude that by using an artificial intelligence system for portfolio selection has performance edge over the human portfolio manager and the market. The systems we selected for this study are only one among numerous artificial intelligence systems available. We would like to conduct further study to better qualify and quantify various artificial intelligence systems for use in the portfolio selection domain.

9. References

[1] Angstenberger, J. Prediction of the S&P 500 Index with Neural Networks. Neural Networks and Their Applications. John Wiley & Sons Ltd. New York, NY., 1996.

[2] Chang, K. C. and R. Fung, "Refinement and coarsening of bayesian networks" In the Sixth Conference on Uncertainty in Artificial Intelligence, 1990, pp.475-482.

[3] Cover, T. M. "Universal portfolios" Mathematical Finance, vol. 1, 1991, pp. 1-29.

[4] Cover, T. M. and D. H. Gluss "Empirical Bayes stock market portfolios", Advances in Applied Mathematics, vol. 7, 1986, pp. 170-181.

[5] Friedman, N., "The Bayesian Structural EM Algorithm", In the Fifteenth International Conference on Machine Learning, 1998.

[6] Freund Y. (1995). Boosting a Weak Learning Algorithm by Majority. Information and Computation, 121(2):256–285, 1995.

[7] Horvitz, E. J. and M. Barry, "Display of Information for Time-Critical Decision Making", In the Eleventh conference on Uncertainty in Artificial Intelligence, 1995, pp.296-305.

[8] Howard, R. A., "Influence to Relevance to Knowledge. Influence Diagrams", Belief Nets and Decision Analysis, 1990, pp. 3-23.

[9] John, G.H., Miller, P. and Kerber R., "Stock Selection Using Rule Induction", IEEE Expert: Intelligent Systems and Their Application, 1996, pp. 52-58.

[10] Jones, C.P., Investments: Analysis & Management. John Wiley & Sons Inc., New York, NY.,2003.

[11] Laskey, K. B. and S. M. Mahoney, "Network fragments: Representing knowledge for constructing probabilistic models", In the Conference on Uncertainty in Artificial Intelligence, 1997, pp. 334-341.

[12] Nickerson, R. C. and D. W. Boyd, "The use and value of models in decision analysis", Operation Research, vol. 28, 1980.

[13] Poh, K. L. and E. Horvitz, "Reasoning about the Value of Decision Model Refinement: Methods and Application", In the Ninth Conference on Uncertainty in Artificial Intelligence, 1993, pp. 174-182.

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

[15] Reilly, F.K. and Norton, E.A., Investments, Chapter 10. South-Western, Mason, OH., 2003.

[16] Ross Quinlan. Data Mining Tools See5 and C5.0. URL http://www.rulequest.com/see5-info.html.

[17] Robert E. Schapire. A Brief Introduction to Boosting. In IJCAI, pages 1401–1406, 1999. URL citeseer. nj.nec.com/schapire99brief.html.

[18] Russell, S. J. and E. H. Wefald, "Principles of metareasoning", Artificial Intelligence, 49: 1991, pp. 361-395.

[19] Starzyk, J. A., Zhen Zhu, H. He, and Zhineng Zhu, "Self-Organizing Learning Array and Its Application to Economic and Financial Problems", In The 3rd International Workshop on Computational Intelligence in Economics and Finance, in Conjunction with the 7th Joint Conference on Information Science, 2003.

[20] Strong, R.A., Portfolio Construction, Management and Protection. Thomson Learning, Chapter 15, 24, 25. Cincinnati, OH., 2000.

[21] Sycara, K. P. and K. Decker, Intelligent Agents in Portfolio Management. Agent Technology, Springer-Verlag, 1997.

[22] Trippi, R.R., Turban, E., Investment Management: Decision Support and Expert Systems. South-Western, Boston, MA., 1990.

[23] Tseng C.C., "Comparing Artificial Intelligence Systems for Stock Portfolio Selection", In The 9th International Conference of Computing in Economics and Finance, University of Washington, Seattle, Washington, 2003.

[24] Tseng C.C., "Influence Diagram for Investment Portfolio Selection", In The 3rd International Workshop on Computational Intelligence in Economics and Finance, 2003.

[25] Tseng, C.C., P. J. Gmytrasiewicz and C. Ching, "Refining Influence Diagram For Stock Portfolio Selection", In the Seventh International Conference of the Society for Computational Economics, 2001.

[26] Watson, S. R. and R. V. Brown, "The valuation of decision analysis", Journal of the Royal Statistical Society, vol. 141, Part I, 1978, pp. 69-78.