ELSEVIER

Self-organizing learning array and its application to economic and financial problems

Z. Zhu [a,*], H. He [a], J.A. Starzyk [a], C. Tseng [b]

[a] *School of Electrical Engineering and Computer Science, Ohio University, Athens, OH 45701, USA*
[b] *Department of Computer Science and Information Systems, Texas A&M University – Commerce, Commerce, TX 75429, USA*

Abstract

A new self-organizing learning array (SOLAR) system has been implemented in software. It is an information theory based learning machine capable of handling a wide variety of classification problems. It has self-reconfigurable processing cells (neurons) and an evolvable system structure. Entropy based learning is performed locally at each neuron, where neural functions and connections that correspond to the minimum entropy are adaptively learned. By choosing connections for each neuron, the system sets up the wiring and completes its self-organization. SOLAR classifies input data based on weighted statistical information from all neurons. Unlike artificial neural networks, its multi-layer structure scales well to large systems capable of solving complex pattern recognition and classification tasks. This paper shows its application in economic and financial fields. A reference to influence diagrams is also discussed. Several prediction and classification cases are studied. The results have been compared with the existing methods.
© 2006 Elsevier Inc. All rights reserved.

Keywords: Self-organizing learning array; Computational intelligence; Economic and financial problems; Stock investment classification and prediction; Bankruptcy; Credit card and loan decision

1. Introduction

Information systems are used in data mining and intelligent decision support to automatically represent the knowledge that can be extracted from databases [15]. Data mining applications have used inductive learning [18] algorithms to extract the rules and futures inferred from data, which include decision trees [17], logic programming [13], and ensemble learning. Inductive learning is deterministic and supervised. It generalizes hidden rules or recovers unknown functions from observation of examples. A decision tree is a simple but effective form of inductive learning. It makes decisions from a set of discrete or continuous attributes. Inductive logic programming performs knowledge-based inductive learning expressed in first-order logic. It can learn the rational knowledge that the attribute-based systems have difficulty obtaining. Ensemble learning selects a

collection of single classifiers and combines their decisions. One of the best-known ensemble algorithms is boosting [12], which is designed to boost the accuracy of individual learning algorithms. In many approaches, artificial neural networks (ANNs) were used for data mining and knowledge extraction in large data sets. An ANN processes information by simulating biological neural systems and has been one of the most popular statistical learning methods. It forms complex nonlinear functions with many parameters, which can be learned via training [18]. Recently, kernel-based learning machines have been proposed, of which the support vector machine (SVM) [4] is considered the most popular technique. An SVM provides optimal separation in the kernel function-induced feature space. It has also been widely used as a powerful tool for classification and regression. These various forms of computational intelligence were used in many specialized fields.

Over the last decade, computational intelligence has been increasingly used to solve difficult economic and financial problems, including modeling, prediction, recognition, and analysis. Financial datasets are usually small-sized with high dimensionality, and contain both quantitative and qualitative attributes. The attributes are often highly correlated and interactive [5]. A lot of effort has been spent to form a bridge between the computational finance and machine learning tools developed by engineers and scientists. McNelis [11] discussed the application of ANNs coupled with evolutionary computation in financial and economic prediction problems. Especially, he demonstrated examples of quantitative forecasting. Although the use of ANNs for decision making can achieve a high predictive accuracy rate, it lacks explanation capability and the reasoning behind how decisions are reached. Baesens [2] presented the results from analysis of real-life credit-risk data sets using neural network rule extraction techniques. Clarifying the ANN's decisions by explanatory rules based on learned knowledge can help credit-risk managers explain why a particular applicant is classified as good or bad. He applied ANN rule extraction and decision tables based on the PROLOGA software to advanced decision-support systems for credit-risk evaluation. Tsitsiklis and Van Roy [24] introduced and analyzed a simulation-based approximate dynamic programming method for pricing complex American-style options, with a possibly high-dimensional underlying state space. This research involves the evaluation of value functions at a finite set, consisting of "representative" elements of the state space. Magdon-Ismail [10] gave a self-contained introduction to the risk neutral or martingale approach to the pricing of financial derivatives. This approach provides a rich source of problems ideally suited to the application of Monte Carlo methods. Finally, Atiya [1] conducted an extensive study and gave a survey of bankruptcy prediction methods. He introduced financial ratio and equity-based indicators for neural network based bankruptcy prediction with improved performance. His work demonstrated that neural networks could deliver a superior performance over other techniques in bankruptcy prediction.

An automatic re-configurable learning machine self-organizing learning array (SOLAR) has been proposed recently [20]. It proved to be a useful classification and prediction tool applicable to different real world problems. SOLAR is a regular, two or three-dimensional array of identical processing cells (neurons), with dynamically re-configurable connections. In our previous work, SOLAR was simulated on standard benchmarks and proved to be advantageous [21] over many existing neural networks and machine learning algorithms. SOLAR can be realized on both software and hardware platforms [22]. In this paper, SOLAR is simulated in software and applied to solve economic and financial problems. Due to its resemblance to the ANNs and SVM, we pay particular attention to their performance compared with SOLAR. This paper is organized as follows. Section 2 briefly describes the SOLAR architecture and single neuron functionality. Section 3 discusses the approach of SOLAR data processing and classification. In Section 4, SOLAR is applied to two stock investment decision support problems, a bankruptcy prediction and two financial status recognition problems. SOLAR's performance is compared with the existing methods. Finally, a summary is given in Section 5.

2. SOLAR architecture and single neuron functionality

A SOLAR architecture is defined by an array of identical cells. Each cell in the array has ability to self-organize by adapting its functionality in response to information contained in its input signals. Cells choose their inputs from neighboring cells and send outputs to other cells. This may result in a multi-level hierarchical system capable of statistical learning, associative memories and reinforcement based self-organization. The SOLAR implementation presented in this paper employs a 2-dimensional feed forward (FF) structure with all neurons arranged in multiple layers. Neurons are connected to the original inputs or the outputs of neurons

from previous layers. Based on the dimensionality and complexity of input data, a variable number of layers and neurons are used, which is automatically decided by the learning algorithm. Typically, the number of neurons per each layer is set equal to or greater than input data dimensionality. For simplicity of implementation, a fixed number of neurons are added per each layer and trained in parallel. All neurons inside the array are pre-wired with the same number of redundant data and control connections, which are pseudo randomly generated. The training procedure refines the connections and establishes the final SOLAR wiring structure. Each neuron in the array has the ability to self-organize by adapting its functionality in response to information contained in its input signals. Similar to the behaviour of SVM, a SOLAR neuron generates a hyper-plane in the input space and separates it into subspaces. But unlike an SVM, hyper-planes are generated concurrently in each neuron's input space. Information from all neurons is merged to form the final decision, which is found analogous to the maximum ratio combination or the boosting algorithm.

It is believed that biological neurons tend to have local connections [6]. Therefore, in SOLAR there is a higher probability that a neuron connects to close neighboring neurons. Some more distant connections are randomly used in the pre-wiring stage with smaller probability. An example structure of a 4-layer SOLAR is shown in Fig. 1, where the circles represent neurons and the triangles are outside inputs. Solid lines stand for data connections and the dashed lines are for control connections.

SOLAR neurons are event-driven processors responding to their selected data and control inputs. Each neuron N receives an input vector x_i and transforms it to a scalar output x_o. In addition, each neuron has a binary control input τ_i and two control outputs τ_o and $\bar{\tau}_o$, as illustrated in Fig. 2.

A logic high signal in the control input activates this neuron. Control outputs are generated and statistical information is obtained when the neuron is activated. With control input high, this neuron only reacts to data from a selected part of the whole input space, which forms a local input space of this neuron. Thus a control input plays the role of inhibitory connections in biological neurons, preventing a controlled neuron from firing. The first column neurons have their control inputs set to 1, so they are always activated. Each neuron
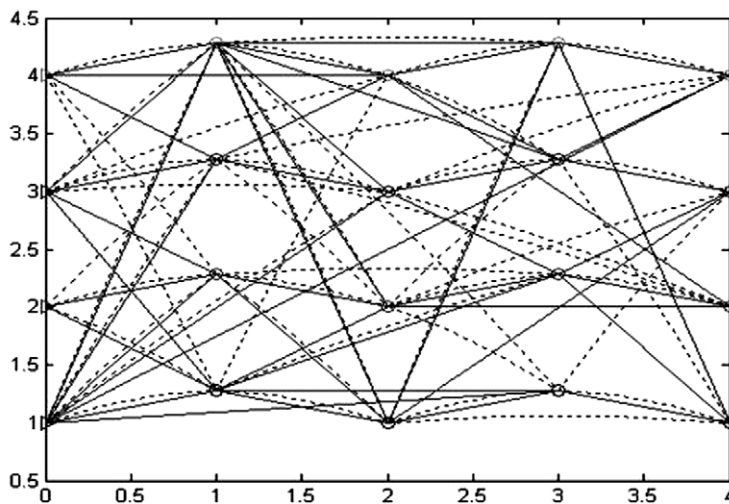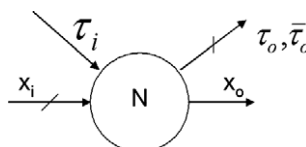

Fig. 1. Pre-wired neuron structure.

Fig. 2. Neuron input and output signals.

performs a simple operation like adding, subtracting and shifting, or a simple approximation of the logarithm and exponential functions. Concatenation of these different basic operations, which results from signal processing by several neurons, yields more complicated transformation of the input space. A neuron generates partition of its input space by comparing its output S_o against a set threshold t, which generates a complex $n - 1$ dimensional manifold to separate an n-dimensional input feature space into two subspaces S and S_i. Complexity of the resulting separation boundary grows along with the increasing number of neuron layers.

Classification performed by each neuron is based on estimated class probabilities in each of the neuron's subspaces. A neuron's transformation functions and thresholds are optimized by contributing neurons to perform a better division of classes using training data. A clear division with a proper threshold provides a strong support for final classification. The succeeding neurons that process the data in one of the subspaces can deal with fewer classes and simpler distribution. The quality of this separation is evaluated based on statistical measures. Searching for the optimum threshold, as well as selecting a neuron's operation, is a training task.

Using the probabilities of training data from different classes that fall into each subspace, information index I can be obtained from

$$I = 1 - \frac{\Delta E_s}{E_{max}} = 1 - \frac{[\sum_{sc} P_{sc} \log(P_{sc}) - P_s \log(P_s)] + [\sum_{sic} P_{sic} \log(P_{sic}) - P_{si} \log(P_{si})]}{\sum_c P_c \log(P_c)}, \tag{1}$$

where P_{sc} is the probability of a class c satisfying threshold, P_{sic} is the probability of a class c not satisfying threshold, P_s is the subspace probability (percentage of input data that passes threshold), P_{si} is the complementary subspace probability (data that does not pass threshold), and P_c is the class probability.

Since neurons are pre-wired with redundant inputs, different combinations of data inputs, transformation operations, and control inputs result in different information index values. Selection of these parameters is a result of local optimization performed by searching for the maximum information index. Instead of using a time consuming search through all the combinations, an efficient binary search algorithm has been designed [22]. In the current implementation, a neuron selects two data inputs out of three pre-wired connections and one control input out of two connections. Thus each neuron has six different input combinations to consider. The neuron also has 32 transfer functions to choose from, each of which is analyzed with an 8-step threshold search. Therefore, (1) is to be computed $6 \times 32 \times 8 = 1536$ times for each neuron, which is an easy job for modern computation hardware. Typically there can be about 30 layers for a complicated classification task with 60 attributes. It costs $60 \times 30 \times 1536 = 2,764,800$ calculations of (1) to train a whole SOLAR network. Since neurons in the same layer can be trained concurrently in parallel computation, the actual time consumed for training is dramatically reduced to the level of $30 \times 1536 = 46,080$ calculations of (1).

Information-based learning enables the SOLAR neurons to perform optimal separation even with high uncertainty in the input space, which is especially desired for financial datasets. The optimized information index is stored together with neuron's configuration data, class probabilities, threshold values, etc. The information index defined by (1) is normalized to $[0, 1]$ interval. When $I = 0$, there was no reduction in data entropy, while $I = 1$ indicates that data entropy in a neuron's input space was reduced to 0. The value of I measures the quality of this neuron's subspace separation. It is related to the definition of information deficiency introduced in (2), which quantifies the amount of information left in the subspaces. Information deficiency is simply a normalized relative entropy value in a local subspace and is defined as follows:

$$\delta_s = \frac{\Delta E_s}{E_{max}} = \frac{\sum_{sc} P_{sc} \log(P_{sc}) - P_s \log(P_s)}{\sum_c P_c \log(P_c)}. \tag{2}$$

Information deficiency helps to self-organize the learning process. A subspace with zero information deficiency does not require any learning—data is well classified in the subspace. An important role of information deficiency is that it can be used as a measure of learning by a group of neurons that work on the same input data.

At the first layer of neurons, it is assumed that the input information deficiency is one. The information index is complimentary to the summation of all the subspace information deficiencies

$$1 - I = \sum_s \delta_s. \tag{3}$$

The information deficiencies for output subspaces are defined as the product of local subspace information deficiencies and input information deficiencies.

$$\overline{\delta}_{os} = \delta_i \delta_s, \tag{4}$$

where δ_i is the input information deficiency. They become input information deficiencies δ_i of the connecting neurons.

δ_i allows a neuron to know if its input subspace has been sufficiently learned. If δ_i is less than or equal to the chosen information deficiency threshold, it indicates that not much information can be gained by further learning.

As subsequent neurons extract information from the input data, there is increasingly less independent information left in the data. The learning array grows by adding more neurons until the information deficiency in the subsequent neurons fall below a set threshold value.

3. SOLAR data processing and classification

3.1. SOLAR data processing

An application input data was presented to SOLAR with n input features or attributes. These n features form the dimensions of the input space. So the jth individual input appears as an n-dimensional vector: $X^j = [X_1^j \quad X_2^j \quad \ldots \quad X_n^j]^T$. Therefore the whole input data set, which consists of s individuals, could be organized in an input matrix

$$\overline{X} = \{X^1, \ldots, X^s\} = \begin{bmatrix} X_1^1 & X_1^2 & \ldots & X_1^s \\ & \ldots & \\ X_n^1 & X_n^2 & \ldots & X_n^s \end{bmatrix}. \tag{5}$$

As often happens in financial datasets, not all the features are continuous numerical values. Some of them may be in the form of discrete symbolic values. Since SOLAR operations accept only numerical inputs, the symbolic features need to be transformed into real numbers [9]. In practical data there may be a few undetermined elements in the input space. A default value needs to be assigned for each of these missing elements to make the input space complete. The desired default values minimize the Mahalanobis distances to the cluster data from the same class as the sample with missing value, as discussed in detail in [21]. Since all the features X_1–X_n are obtained from possibly different measurements, their scale may vary greatly. To equalize their significance, all the input features have to be rescaled.

As the result, the pre-processing of SOLAR's input matrix is carried out in three steps:

1. Make all the features numerical, and set values for symbols of the discrete features.
2. Determine the default values for each feature, and use them in place of the missing features.
3. Rescale all the features to a unified range.

Although the processing speed of SOLAR can benefit from smaller dimensionality of the input data, it is not necessary to manually select the important attributes from the dataset based on expert knowledge. SOLAR is able to find correlation between features and make classification decision using all the relevant ones automatically, based on the information contained in the input space. Automatic feature selection techniques are also developed to further improve the efficiency of learning. Similar training performed on large dimensionality data sets could be very expensive for ANN processing, both in computing time and hardware implementation cost.

3.2. SOLAR classification

Data to be classified is sent to the SOLAR network, and the network performs classification based on its training results. As a result of training, neurons internally save the correct recognition probabilities of all the

classes for both of the output spaces as two probability vectors $[P_{sc}]$ and $[P_{sic}]$, where c stands for different classes. If an input data point falls in a voting neuron's input subspace, the neuron is going to vote for this data using its estimated probabilities P_{sc} or P_{sic} as probabilities of correct classification P_{cc} for that class. Each individual neuron's confidence in its classification is only P_{cc}, which can be considered as "weak learning". The voting mechanism gathers all the information and classifies the input signal using a weight function designed after maximum ratio combination (MRC) technique used in mobile communication [16].

$$B_c = 1 - \frac{1}{1 + \sqrt{\sum_{i=1}^{n} \left(\frac{1}{1/P_{cc_i} - 1 + \varepsilon} \right)^2}}, \tag{6}$$

where $P_{cc_i} = P_{cc}$ of each "vote" for class c; $n =$ number of voting neurons; $\varepsilon =$ a small number preventing division by zero.

This weight function provides a statistically robust fusion of individual neuron's votes. It is in fact an ensemble learning algorithm that improves the accuracy of weak learning. The classifier chooses a class c with the maximum weight B_c.

3.2.1. Example

Assume that there are five neurons in a classification problem with three classes, and that the class probabilities estimated by each neuron at the learning stage are as shown in Table 1 (the sum of probabilities in each column is equal to 1). The largest probability value for each class is also listed, assuming that a winner-takes-all voting scheme is used.

Using $\varepsilon = 0.001$ in (6), the weights of different classes for this particular input dataset are calculated and shown in Table 2.

Based on this result, SOLAR will classify this input as a sample from class 2. Notice that if the "winner takes all" approach was used, as shown in Table 1, voting neuron No. 3 should make a decision and this particular input would be classified as class 3.

Table 1
Class probabilities in neurons' output spaces

	Voting neuron number					
	1	2	3	4	5	Winner-takes-all
Class 1	0	0.293	0.079	0.671	0.305	0.671
Class 2	0.753	0.632	0.125	0.329	0.695	0.753
Class 3	0.247	0.075	*0.796*	0	0	*0.796*

Table 2
Voting weight B_c for different classes

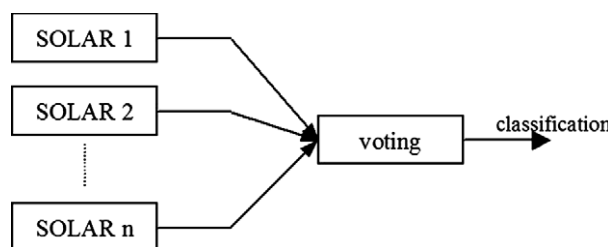| Class 1	Class 2	Class 3
0.6800	*0.8075*	0.7960

Fig. 3. SOLAR ensemble.

As mentioned above, the initial structure of SOLAR is randomly generated. The diversity of individual systems may result in different classification results on the same testing set. Although classification performance of a SOLAR network can be assured using (6), an ensemble of multiple SOLAR networks can be used to obtain stable, statistically robust output. Several SOLAR systems with random differences in their initial structures are generated in parallel. They are trained separately and vote together on the same testing set. It is, in fact, a simple version of ensemble learning, where a single SOLAR implements "weak learning". An example of a SOLAR ensemble is shown in Fig. 3, where *n* individual SOLARs are combined to perform classification.

4. SOLAR application to economic and financial problems

In this work SOLAR software has been applied to economic and financial problems and compared to specialized financial analysis learning methods, which use expert knowledge and are optimized for particular problems. The selected reference approaches are using influence diagrams, ANNs, SVMs, decision trees and other methods.

4.1. Influence diagram

Before applying the SOLAR concept to market investment decision support, the influence diagram approach was first used [23] for reference. We discuss this approach in some detail to point out the differences in the procedural approach to the problem formulation and its solution. Most noticeably, expert knowledge is required to define the initial structural organization of the diagram that can be refined using a machine learning technique.

An influence diagram is a special type of Bayesian network (Fig. 4), 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, shown as ovals, represent random variables in the environment. Decision nodes, shown as squares, represent the choices available to the decision-maker. Utility nodes, either 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 depending on their destinations. Dependency arcs are the arcs that point to the utility or chance nodes representing probability or functional dependence. Informational arcs are the arcs that point
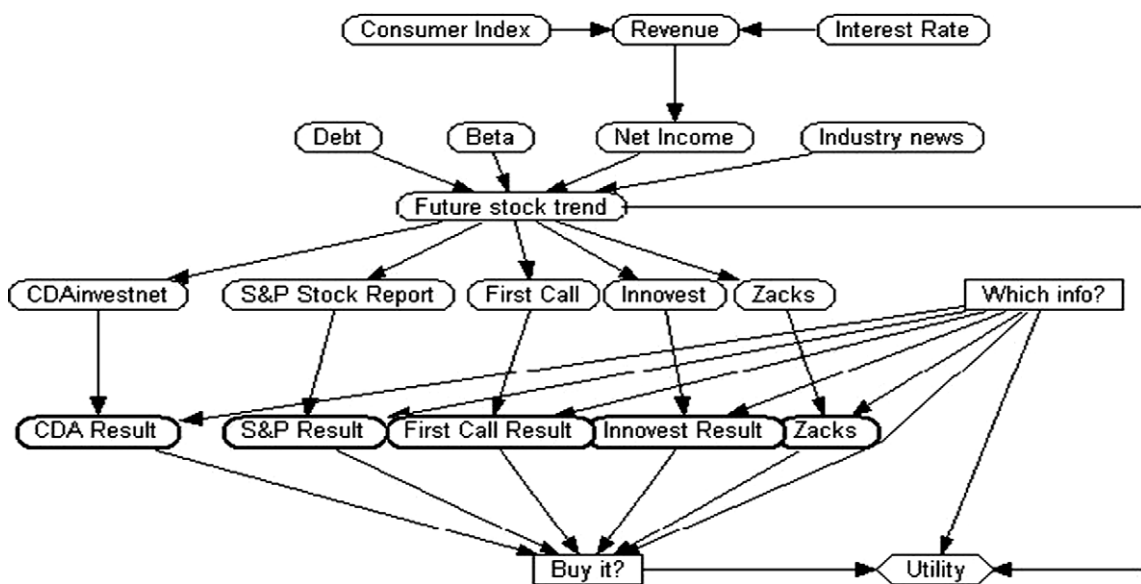

Fig. 4. A simple influence diagram.

to the decision nodes implying that the pointing nodes will be known to the decision-maker before the decision is made.

There are some fundamental characteristics of the influence diagram that one must take into consideration when using one for decision support problems. These characteristics influence the data requirements and 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 to this problem. Some suggest learning from the user's behavior, while 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 knowledge about the structure, probabilistic knowledge for prior and conditional probabilities. There are many variables in a specific problem domain and there might exist several concepts in the problem domain that are observationally equivalent, which means they are not distinguishable even with infinite data. Finding out which of those are relevant to the problem and their casual relationships present a challenge to the knowledge engineer. A significant amount of research and many software tools were devoted to learning of the model structure from data [7]. There are two methods to obtain the probability distribution for a node. First, the probability distributions can be based on frequency of occurrence 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.

4.2. Case studies

Several application examples were selected for testing and comparative study. Cases 1 and 2 are the cases of decision support for stock market investment. SOLAR is used to choose profitable companies. In case 1, the learning results of SOLAR are compared with the outcome of a special type of Bayesian network, the influence diagram designed for stock investment decision support, and the SVM algorithm. It is also compared with the SVM in case 2. Case 3 is the bankruptcy prediction problem, where SOLAR is used to predict companies that will file for Chapter 11 within 3 years. Results obtained in this case are compared with those obtained with Atiya's method [1]. Case 4 is the credit card approval decision. SOLAR made decisions on whether or not to approve a credit card based on personal information. Case 5 is the adult income classification problem. Using these classification results, banks are able to make loan decisions. In both cases 3 and 4, SOLAR is compared to the benchmark approaches reported in literature.

Case 1: S&P 500 Stock investment decision support

The first case study is based on S&P 500 index companies from 1993 to 2003. The problem is to classify these companies into two portfolios and select the one expected to be more profitable. Both networks, SOLAR and the influence diagram were trained on the 1993–2002 data obtained from the CompuStat and tested on the 2003 data. This data contained the company's 1 year return and other financial information, including market/book value, ROE, earning before tax and interest, pretax profit margin, total asset turnover, DOE, financial leverage index and Beta. The performance metric is the average annual return rate from the selected portfolio in the year of 2003, by percentage.

The initial structure of the influence diagram was constructed by consulting with the domain expert. Any such model is inevitably a simplification of what the expert knows, which itself is a simplification of the real-world problem. The essential issue is to decide which variables and relationships are important, and which are redundant and can be omitted. After several interviews with the domain expert, we came up with the portfolio
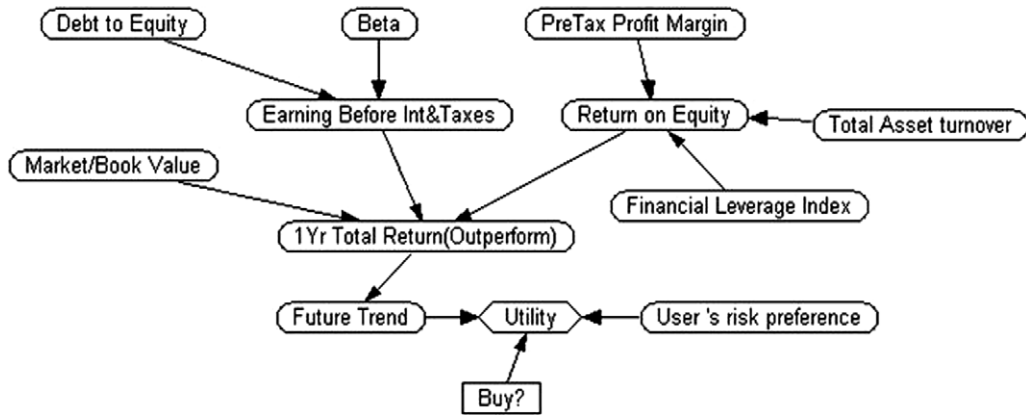
Fig. 5. Influence diagram structure for stock portfolio selection.

selection model shown in Fig. 5. We then applied our heuristic guided refinement algorithm [23] to the influence diagram. The algorithm uses mutual information as a guide to refine the model in order to increase the model's performance.

Once the initial diagram was constructed, the next step was to define the number of values for each variable. Most of the variables are continuous, but all were modeled as discrete. For the first prototype, we modeled the diagram as simply as possible, and each value of each variable was carefully set. For example, originally there were two values in the Beta node, which were increased into a range of values after refinement. Such explicit setting of variables is necessary to avoid ambiguity when assessing conditional probabilities. We applied the refinement algorithm to only the financial factor change nodes (Beta, ROE, etc.), and the risk tolerance remains unchanged for this experiment.

On the 2003 test data, the influence diagram portfolio obtains its maximum average annual return performance when the portfolio contains 134 companies out of the 500. The 134 companies produced an average 1-year total return of −10.2%, while the average return of all 500 companies was −17.8%.

Next, SOLAR was used to analyze the same data. The training data was divided into two classes, with their annual return below or above average respectively. SOLAR was constructed without expertise in this field and has not been refined for this specific problem. However, the self-organizing structure helps the system to automatically extract the useful information. As afore stated, single networks of SOLAR may provide various results and the ensemble of multiple networks improves its stability. Nine individual SOLAR networks were used in this case, each of which yielded −9.2% to −10.5% independently. The whole system chose 329 companies and the average return was −9.9%. Since SVM is usually considered as the optimal classifier, it compared with the performance of SOLAR. A library of SVM can be found in [3]. We used the generic C-SVM with linear, polynomial and RBF kernels, of which a fifth degree polynomial gave the best performance at −10.05%. It is recognized that an optimal selection of SVM type, kernel function and parameter setting may bring better results.

Case 2: Research insight stock investment decision support

We expect SOLAR to be used as a strategic decision support system in a lot of applications besides classification. Therefore, we also applied SOLAR to a stock price prediction problem and compared the results to the optimal classifier SVM. We used Research Insight [19] a financial database derived from over 10,000 publicly traded US companies and closed-end funds trading on the NYSE, AMEX, NASDAQ, OTC and Canadian stock exchanges. Sixty-four financial data indicators were reported by these companies over the most recent 20 years including income, balance sheet, and cash flow statements. The training and testing datasets were constructed based on the 192 features extracted from the database over a 3-year period. Similarly to the annual return classification, two classes were defined for this problem based on financial performance, measured as 1-year stock price change right after the 3-year period. Companies whose stock price increase

was less than the median value of the whole set were classified as class 1 and those above the median were class 2. By using this classification scheme (based on the price increase) we effectively predict future financial performance without explicit reference to time prediction used in the time series approach. The 3-year period of each training dataset was used to develop a classifier based on this time frame.

First we separated the companies that have missing features from the complete ones. Due to the high dimensionality of the dataset and correlation between features, QR decomposition was used to reduce redundancy. After that, missing data were recovered from the complete and independent features. Fig. 6 shows the block diagram of the whole process.

Using the processing shown in Fig. 6, we obtained the cross-validation of the classifiers performance on various datasets. Table 3 shows the analysis results using SOLAR and SVM based on the companies with complete data in both training and testing sets. Each row represents a classifier trained with data from different years and is named as C2000, C2001, C2003. Each column shows the correct classification rate tested on the data from different years. For instance, classifier C2000 was developed based on data slice from 1998 to 2000 and used the stock price change reported in 2001 for classification. It was applied to predict the performance in 2001–2003. Classification results above 50% indicate that we can obtain a better than average performance. The average prediction rate and its standard deviation for each classifier are calculated. As we can see from Table 3, both SVM and SOLAR can reach above 50% performance. SOLAR showed the ability to generalize the hidden rule of forecasting profitable companies from financial information.
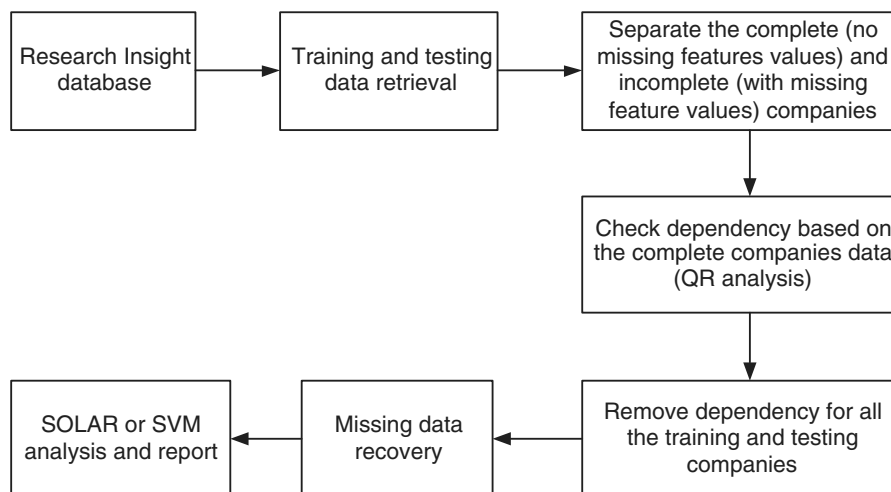

Fig. 6. Research insight data analysis procedure.

Table 3
SOLAR analysis results for research insight data set

Classifier		Testing year					
		2000	2001	2002	2003	Average	Std.
SOLAR	C2000	–	0.62017	0.60718	0.5396	0.5890	0.0433
	C2001	0.54673	–	0.54454	0.51568	0.5357	0.0173
	C2002	0.5673	0.58083	–	0.50928	0.5525	0.0380
	C2003	0.52755	0.49051	0.49908	–	0.5057	0.0194
SVM	C2000	–	0.58405	0.56625	0.57305	0.5745	0.0090
	C2001	0.56197	–	0.60875	0.46193	0.5442	0.0750
	C2002	0.54915	0.57265	–	0.4784	0.5334	0.0491
	C2003	0.60684	0.5584	0.62	–	0.5951	0.0324

Case 3: Bankruptcy prediction

Bankruptcy prediction has been a very important topic in the past decades attracting a lot of attention and research effort. Atiya gave a comprehensive review of existing bankruptcy prediction methods and provided results of his study using traditional neural networks [1]. He collected data for 716 solvent US corporations and 195 defaulted (defaulted within 1–36 months). Taking different instances of a company financial status before the default, he expanded the data set to 1160 input points. These points are grouped as an in-sample set and an out-sample set, used for training and testing respectively. There are two systems of indicators discussed in that paper. One is based on financial ratios alone (called the financial ratio system), containing book value/total assets BV/TA, cash flow/total assets CF/TA, rate of change of cash flow per share, ROC(CF), gross operating income/total assets GOI/TA, and return on assets ROA. The other is based on financial ratios and price-based indicators (equity-based system), with book value/total assets BV/TA, cash flow/total assets CF/TA, price/cash flow ratio P/CF, rate of change of stock price ROC(P), rate of change of cashflow per share ROC(CF) and stock price volatility VOL. Using both of these systems of indicators he obtained a significant improvement in the bankruptcy prediction compared with traditional bankruptcy prediction techniques. His success was mostly based on a proper (expert) choice of indicators.

We used SOLAR to perform bankruptcy prediction based on the input space data of the financial ratio and equity-based system. The correct rates are compared with the results from [1] on the out-of-sample set in Table 4, shown as the results of selected indicators. In this case SOLAR gives comparable results to the method presented in [1], comparing the second and third columns in Table 4.

Since SOLAR is able to handle complicated problems, we also carried out this prediction with all available indicators (63 in our database) instead of using the selected indicators system introduced in [1]. A single SOLAR network can provide significant improvement in correct prediction rate (90.04% vs. 85.5%), as shown in the fourth column of Table 4. SOLAR is a general-purpose identifier and predictor. It was never designed nor optimized for any particular type of problem. However, this experiment shows that SOLAR is good at prediction problems based on large size databases, since it is able to use all the information contained in every indicator and the most correlated indicators automatically contribute the most in classification.

Case 4: Australian credit card approval problem

Credit card approval is a common problem that most machine learning algorithms can be applied to. To compare SOLAR with other existing methods, a credit card database [14] was used as a benchmark. This database is available from University of California at Irvine—ftp at cs.uci.edu (128.195.1.1) in directory/pub/machine-learning databases. It has 690 instances in two classes and 14 attributes, 6 quantitative and 8 qualitative. Thirty seven instances have one or more features unavailable. Pre-processing [21,9] has been applied to convert the qualitative attributes into numerical ones and to assign default values for the missing parts.

Several traditional classification algorithms have been tested on this benchmark [14] including learning machines, neural networks and statistical methods. Their misclassification rates were reported in the literature and are listed in Table 5 together with results from SOLAR networks. As can be seen from Table 5, SOLAR shows a better classification rate than all the listed methods except for CAL5. Notice, however, that SOLAR

Table 4
Correct prediction rate of bankruptcy

Time to default	Correct prediction rate in %		
	Using [1]	Using SOLAR	
	Selected indicators	Selected indicators	All indicators
6 month or less	86.15	85.11	87.23
6–12 months	81.48	84.09	86.36
12–18 months	74.60	76.19	90.24
18–24 months	78.13	55.17	72.24
More than 24 months	66.67	64.29	75.00
Total defaulted	78.13	75.13	83.96
Solvent	90.07	92.74	93.42
Total	85.50	85.80	*90.04*

Table 5
Misclassification rate comparison on credit card approval

Algorithm	Misclassification rate	Algorithm	Misclassification rate
CAL5	0.131	C4.5	0.155
SOLAR	*0.135*	SMART	0.158
Itrule	0.137	Baytree	0.171
DIPOL92	0.141	AC2	0.181
Logdisc	0.141	k-NN	0.181
Discrim	0.141	NewID	0.181
CART	0.145	LVQ	0.197
RBF	0.145	ALLOC80	0.201
CASTLE	0.148	CN2	0.204
Naivebay	0.151	Quadisc	0.207
IndCART	0.152	Default	0.440
Bprop	0.154	Kohenen	–

Table 6
Misclassification rate comparison adult income classification

Algorithm	Misclassification rate	Algorithm	Misclassification rate
FSS Naïve Bayers	0.1405	CN2	0.1600
NBTrees	0.1410	Naïve Bayers	0.1612
C4.5-auto	0.1446	Voted ID3 (0.8)	0.1647
IDTM (decision table)	0.1446	T2	0.1687
HOODG/SOLAR	*0.1482*	1R	0.1954
C4.5 rules	0.1494	Nearest-neighbor (3)	0.2035
OC1	0.1504	Nearest-neighbor (1)	0.2142
C4.5	0.1554	Pebls	Crashed
Voted ID3 (0.6)	0.1564		

performs better than all the neural network methods listed in this table. In addition, decision tree methods, such as CAL5 and C4.5 are believed to have better performance on credit card problems [14], while SOLAR was not specifically optimized for this type of problem (Table 6).

Case 5: Loan decision—adult income classification [9,14]

Potential customer analysis is an example of another real world application where banks or financial companies can use the computational intelligence approach for decision-support. The database, which was also obtained from the University of California at Irvine, consists of two sets of data. One set contains the training data and has 32,561 instances of applications, while another one was used as testing data and has 16,281 instances. The dataset contains age, work-class, final weight, education, education-num, marital-status, occupation relationship, race, sex, capital-gain, capital-loss, hours-per-week, and native country, eight of which are symbolic values. Fourteen percentage of the instances have missing data. There are also two classes, class 1— 23.93% people with earnings greater than or equal to 50,000$/year and class 2—76.07% people with earnings below 50,000$/year. The data set consists of a number of instances of each class. This is a problem with complex relationships between the selected features.

Again, performance of SOLAR was compared with the existing methods. Although SOLAR did not perform as well as the best algorithms, it is the only artificial neural network on the list.

5. Summary

A new computational intelligence algorithm—the self-organizing learning array (SOLAR) was described and applied to a number of specialized economic and financial cases. Although such datasets are usually small-sized with high dimensionality, and contain both quantitative and qualitative attributes, the information based learning of SOLAR successfully demonstrated the capability of handling prediction, classification and

recognition in this field. Since SOLAR performs its pattern recognition tasks on subsets of its neurons, it evolves its structural organization representing information included in these subsets according to the locally controlled learning objectives. It can be efficiently implemented in parallel computation and in real time hardware structures. Associative and reinforcement learning are some of its potential features. This is a topic for further study.

Acknowledgements

References

[1] A.F. Atiya, Bankruptcy prediction for credit risk using neural networks: a survey and new results, IEEE Transactions on Neural Networks 12 (4) (2001) 929–935.
[2] B. Baesens, R. Setiono, C. Mues, J. Vanthienen, Using neural network rule extraction and decision tables for credit-risk evaluation, Management Science 49 (3) (2003) 312–329.
[3] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, 2004. Available from: <www.csie.ntu.edu.tw/~cjlin/libsvm/>.
[4] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods, Cambridge University Press, 2000.
[5] V. Dhar, D. Chou, F. Provost, Discovering interesting patterns for investment decision making with GLOWER—a genetic learner overlaid with entropy reduction, Data Mining and Knowledge Discovery 4 (4) (2000) 251–280.
[6] J.E. Dowling, Creating Mind: How the Brain Works, W.W. Norton & Company, Inc., New York, 1998.
[7] N. Friedman, The Bayesian structural EM algorithm, in: Proceedings of the Fifteenth International Conference on Machine Learning, 1998, pp. 129–138.
[8] K.B. Laskey, S.M. Mahoney, Network fragments: representing knowledge for constructing probabilistic models, in: Proceedings of the Conference on Uncertainty in Artificial Intelligence, 1997, pp. 334–341.
[9] T.-H. Liu, Future hardware realization of self-organizing learning array and its software simulation, M.S. thesis, Ohio University, Athens, OH, 2002.
[10] M. Magdon-Ismail, The equivalent martingale measure: an introduction to pricing using expectations, IEEE Transactions on Neural Networks 12 (4) (2001) 684–693.
[11] P.D. McNelis, Neural Network with Evolutionary Computation: Predictive Edge in the Market, Elsevier Academic Press, 2004. Available from: <http://www.georgetown.edu/faculty/mcnelisp/nnga11a.pdf>.
[12] R. Meir, R. Gunnar, An Introduction to Boosting and Leveraging, Advanced Lectures on Machine Learning, Springer-Verlag New York, Inc., New York, NY, 2003.
[13] R.S. Michalski, Inferential theory of learning: developing foundations for multistrategy learning, in: R. Michalski, G. Teccuci (Eds.), Machine Learning: A Multistrategy Approach, vol. IV, Morgan Kaufmann, 1994, pp. 3–61.
[14] D. Michie, D.J. Spiegelhalter, C.C. Taylor, Machine Learning, Neural and Statistical Classification, Ellis Horwood Ltd., London, UK, 1994.
[15] Z. Pawlak, Information systems—theoretical foundations, Information Systems 6 (1981) 205–218.
[16] J.G. Proakis, Digital Communications, third ed., McGraw Hill, New York, 1995.
[17] J.R. Quinlan, Induction of Decision Trees, Machine Learning Journal 1 (1) (1986) 81–106.
[18] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, second ed., Prentice Hall, New Jersey, 2003.
[19] Standard & Poor's Research Insight SM COMPUSTAT (North America): A Primer for Getting Started [Online]. Available from: <http://www2.library.unr.edu/dataworks/compu_primna76.pdf>.
[20] J.A. Starzyk, Y. Guo, Entropy-based self-organized ANN design using Virtex FPGA, in: Proceedings of the 2001 International Conference on Engineering of Reconfigurable Systems and Algorithms, Las Vegas, NV, 2001, pp. 103–106.
[21] J.A. Starzyk, Z. Zhu, Software simulation of a self-organizing learning array system, in: Proceedings of the 6th IASTED International Conference on Artificial Intelligence & Soft Comp., Banff, Alberta, Canada, 2002, pp. 84–89.
[22] J.A. Starzyk, Z. Zhu, T.-H. Liu, Self organizing learning array, IEEE Transactions on Neural Networks 16 (2) (2005) 355–363.
[23] C. Tseng, P.J. Gmytrasiewicz, C. Ching, Refining influence diagram for stock portfolio selection, in: Proceedings of the Seventh International Conference of the Society for Computational Economics, 2001, pp. 241–256.
[24] J.N. Tsitsiklis, B. Van Roy, Regression methods for pricing complex American-style options, IEEE Transactions on Neural Networks 12 (4) (2001) 694–703.