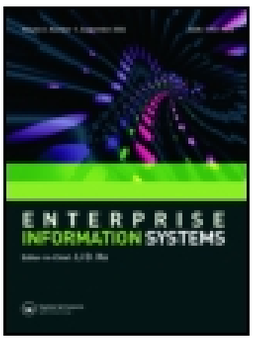

Enterprise Information Systems

ISSN: 1751-7575 (Print) 1751-7583 (Online) Journal homepage: http://www.tandfonline.com/loi/teis20

Design and development of inventory knowledge discovery system

CKM Lee, Catalin Mitrea, W.H. Ip, KL Choy & GTS Ho

To cite this article: CKM Lee, Catalin Mitrea, W.H. Ip, KL Choy & GTS Ho (2016): Design and development of inventory knowledge discovery system, Enterprise Information Systems, DOI: 10.1080/17517575.2016.1221143

To link to this article: http://dx.doi.org/10.1080/17517575.2016.1221143

Published online: 02 Sep 2016.

ENTERPRISE INFORMATION SYSTEMS, 2016
http://dx.doi.org/10.1080/17517575.2016.1221143

Taylor & Francis
Taylor & Francis Group

Design and development of inventory knowledge discovery system

CKM Lee[a], Catalin Mitrea[b], W.H. Ip[a], KL Choy[a] and GTS Ho[a]

[a]Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Hong Kong, China; [b]School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore

ABSTRACT

Inventory management (IM) performance is affected by the forecasting accuracy of both demand and supply. In this paper, an inventory knowledge discovery system (IKDS) is designed and developed to forecast and acquire knowledge among variables for demand forecasting. In IKDS, the TREes PArroting Networks (TREPAN) algorithm is used to extract knowledge from trained networks in the form of decision trees which can be used to understand previously unknown relationships between the input variables so as to improve the forecasting performance for IM. The experimental results show that the forecasting accuracy using TREPAN is superior to traditional methods like moving average and autoregressive integrated moving average. In addition, the knowledge extracted from IKDS is represented in a comprehensible way and can be used to facilitate human decision-making.

ARTICLE HISTORY
Received 9 July 2013
Accepted 2 August 2016

KEYWORDS
Artificial neural network; forecasting; inventory management; knowledge discovery systems; TREPAN algorithm

1. Introduction

Enterprises need to pay attention to the dynamic market and changing customer recruitment so as to formulate strategic inventory management (IM). Companies may hold stock with the intention of selling it or transforming it into a more valuable state. Practitioners usually make decisions based on past experience and knowledge, which is often implicit and vague. The majority of managerial decisions can be described as inference processes, where certain rules of inference under a given situation and apply knowledge are applied in order to achieve a certain goal. Human experts can perform a wide variety of inferences extremely fast, but human resources in an organization are limited and expensive. It was predicted by IDC that 40 zettabytes (ZB) 5200 GB of data was to be generated by every person on earth by 2020 but only 3% of the data will be analysed (IDC, 2012). Apart from the growth in data volume, the data has large variety and increases at a high velocity. It leads to the emergence of big data and research concerns on how to analyse the big data so as to enhance decision-making (Russom 2011). Elgendy and Elragal (2014) reviewed the latest analytical techniques and tools of big data, and it was also realized that big data can enable performance improvement in the supply chain through accurate forecasting and planning. Dhar (2013) examined the importance of data science in extracting knowledge from data with integrated skill sets in machine learning artificial intelligence, statistics and optimization skills. Human experts are unable to analyse all the data that is accumulated in organizational databases, but computers can analyse it with the help with artificial intelligence techniques such as artificial neural networks (ANNs). ANNs are information processing systems which imitate the functionality of the human brain to

CONTACT CKM Lee ckm.lee@polyu.edu.hk Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Hong Kong

analyse data by finding relations and identifying the independent and dependent variables (Harvey 1994; Anderson and Davis 1995).

The objectives of this research are to investigate, design and develop a knowledge discovery and extraction system, with ANN, for inventory forecasting. The significance of this research could lead to improvements and better understanding of the relevance of the parameters used as variables in the forecasting models. ANNs are used to extract knowledge from historical data. Therefore, it is important to understand how a neural network (NN) learns and what topologies are used in the literature for forecasting.

This research paper is organized as follows: Section 2 presents the literature related to IM and the factors affecting demand forecasting accuracy, ANN and rule-extracting techniques are described in Section 3 related to the model of inventory knowledge discovery system (IKDS). In Section 4, the experimental results are shown. The paper concludes with a summary of the findings and contributions in the application of advanced data analysis techniques to IM.

2. Literature review

The literature review begins with an overview of IM and the influential factors which affect forecasting accuracy of the demand. Next, this study addresses the NN as artificial intelligence technique for discovering knowledge hidden within data. The rules extracted from NNs to forecast inventory are addressed. Finally, the implication of the research is discussed based on the review findings.

2.1. *Inventory management*

IM is essential for almost all firms within the supply chain. Yelland (2010) adopts the Bayesian method to predict the demand of parts of high technology products using hierarchical priors. To achieve high performance in IM, factors which influence the performance are to be identified as technical factors, organizational factors, financial factors, managerial factors and informational factors. Lieberman, Helper, and Demeester (1999) realized the inventory performance can be grouped into the following: (1) safety stock (SS) planning of the enterprise, (2) capacity utilization level, (3) purchasing effectiveness, (4) demand forecasting accuracy, (5) level of IM practices pursued, (6) management attitude, (7) employees training, (8) interaction with suppliers, (9) interaction with customers and (10) supplier empowerment and collaboration (Li 2012). In general, researchers (Yan, Ji, and Wang 2012; Qi 2013) have identified many factors that influence inventory performance such as the lead time, supply reliability, delivery reliability and forecasting accuracy. Forecasting accuracy is one of the most important influencing factors for the SS, and therefore of the IM, as it is determined by external factors such as the number of customers and customer satisfaction, or internal factors such as product quality and reliability.

2.2. *Demand forecasting*

Demand forecasting is an essential process for managing inventory efficiently. A well-designed forecasting system can contribute significantly to improve the supply chain performance, especially in IM which is dependent on the accuracy of the forecasting method. The number and variety of available forecasting methods and techniques makes the choice of the most appropriate method a complex process. According to Chopra and Meindl (2007), forecasting methods are classified as qualitative and quantitative: (1) Qualitative forecasting methods are subjective and rely heavily on human expert judgment; (2) quantitative methods are objective and are based on quantitative models which rely heavily on mathematical computations. Quantitative methods include simulation models, time series models and causal models. Simulation forecasting methods imitate the customer choices when purchasing a product.

Time series methods (Wang and Jain 2003; Abdel-Aal 2008; Box, Jenkins, and Reinsel 2016), which use historical records of demand to make a forecast, are based on the assumption that past demand history is a good indicator of future demand. Time series methods can be used to identify systematic and seasonal variations in the data, cyclical patterns, trends and growth rates of the trends. Appropriate methods include autoregressive (AR), AR with external input series, vector autoregression (VAR), autoregressive moving average (ARMA) and autoregressive integrated moving average (ARIMA). In some cases, the variable to be forecast has a rather direct relationship with one or more of the other variables whose values are known at the time of the forecast. The rationale behind the causal methods is to use refined and specific information concerning the variables so as to develop a relationship between a lead event and the event being forecasted (Bowersox, Closs, and Cooper 2002). Another alternative forecasting approach is ANN which shows promising performance compared with the traditional forecasting techniques.

2.3. *Forecasting with an artificial neural network*

An ANN is a parallel information processing system that has certain functionalities and performance characteristics in common with biological neuron networks. Two types of NNs are adopted in this work: (1) feedforward neural network (FNN) and (2) non-linear autoregressive exogenous neural network (NARX). The use of FNN in forecasting can be described intuitively as follows: Given a certain amount of historical data which can be used to analyse the behaviour of a particular system, such data can be used to train ANNs to correlate the system with respect to time or other system parameter. Even though this seems a simplistic mechanism, experience shows that the ANN approach is able to provide a more accurate prediction than expert systems or statistical counterparts (Carbonneau, Laframboise, and Vahidov 2008; Carmo and Rodrigues 2004; Chang, Wang, and Tsai 2005; Gutierrez, Solis, and Mukhopadhyay 2008). NARX (Menezes and Baretto 2006, Lin et al. 1996, Siegelmann, Horne, and Giles 1997) is a recurrent neural architecture commonly used for input–output modelling of non-linear systems. The input of NARX is formed by two tapped-delays, one sliding over the input signal and other over the output signal. In most past studies, the ANN is treated as a black box because it lacks the explaining capabilities of the learnt models. This is the reason why different methods to extract knowledge embedded in trained ANNs have been developed.

2.4. *Rule extraction from trained neural networks*

In recent research work, most rule-extracting techniques are restricted to represent the networks description as a set of IF-THEN-ELSE rules. This group of algorithms can be separated into *decompositional, pedagogical and eclectic approaches* (Augasta and Kathirvalavakumar 2012). (1) The decompositional approach extracts rules at the level of individual (hidden and output) units of the trained network. The output from each hidden and output unit is mapped into a binary outcome that corresponds to a rule consequent. Since each hidden or output unit can be interpreted as a step function, only situations in which the summed input of a set of incoming links guarantees the activation of the unit has to be determined to construct local rules for the respective unit. Examples of these methods are the Kernighan-Lin algorithm developed by Fu (1991) and the SUBSET algorithm developed in Towell and Shavlik (1993). (2) The pedagogical approaches treat the network as a black box. The rule extraction process is constructed as a learning task and uses the training network just to generate examples. Some examples of such methods are the VI-analysis techniques developed in Thrun (1993), which use validity intervals to derive the rules. (3) The eclectic approaches, which combine decompositional and pedagogical methods (Hruschka and Ebecken 2006; Towell and Shavlik 1993), focus on extracting rules that include the most important input units. The TREes PArroting Networks (TREPAN) algorithm (Torres

and Rocco 2005; Craven and Shavlik 1996), which does not require specialized ANN architecture or training, meets these requirements. This algorithm is so flexible that it is not restricted to FNN but can be applied to other systems, including standard statistical classifiers. Moreover, TREPAN builds an *M*-of-*N* based decision tree that represents the function that the network has learned by recursively partitioning the input space. The output of TREPAN is a decision tree. The topology of the tree is represented in terms of nodes, parents, children, leaves and forks.

2.5. *Implications from the literature review*

In summary, the review of contemporary publications indicates that whilst many research studies have been conducted using various approaches to discover knowledge, research related to knowledge discovery for IM has not yet received the *attention* it deserves. In the literature review, the influential factors of IM and SS have been identified, but there is not enough study in depicting which factors are more important in comparison to others. To solve these issues, the ANN is regarded as a promising technique for knowledge discovery for classification, clustering and prediction, but it lacks explanation ability. It would be advantageous to explain knowledge from the trained NNs with decision trees and rules, which are comprehensible forms of knowledge representation. Probably, the most important reason for extracting decision trees and rules from an ANN is to facilitate knowledge acquisition. As observed in Craven and Shavlik (1996), 'a learning system may discover salient features in the input data whose importance was not previously recognized'.

The next section proposes a research methodology which integrates steps from fields such as knowledge discovery and deals with data acquisition and preparation. ANNs are used to map inputs with output variables in the learning process (in a prediction task), and rule extraction algorithms are adopted to understand the models learnt by networks.

3. Model of inventory knowledge discovery system

This section begins by proposing an IKDS framework which is the extended work of Lee et al. (2011) on a responsive logistics workflow system where the previous work of Lee et al. (2011) focused on recognizing demand patterns by NN and formulate the replenishment strategy. Since most researchers have realized that ANN is like a blackboard with known input and output, detailed explanation of the process behind the knowledge extraction is ignored. This paper is more concerned about explaining the demand forecast result with the TREPAN algorithm when extracts the knowledge learnt by a trained ANN in the form of decision trees. The proposed IKDS shown in Figure 1 is composed mainly of three modules: data preparation module, NN module and TREPAN module. The framework is composed of 15 steps, from 1 to 15. Regarding the flow of information, the framework has two inputs (goal definition and database) and one output (use of knowledge). The data preparation module deals mainly with the selecting and preprocessing of the raw data contained in a database. The NN module is used to learn hidden relations in the data used to forecast, and the TREPAN module is used to extract the knowledge acquired and embedded in NN.

3.1. *Data preparation module*

The data preparation module includes data acquisition, data selection, data preparation and data transformation. (1) Data acquisition deals with searching, identifying and purchasing real data contained in inventory databases from different industrial players and companies (Figure 2). Data can be acquired in the form of electronic tables and/or questionnaires and have different formats like excel, mysql, sql-server, oracle and simple text tables. If the tables or questionnaires are available in hard copy, transformation in an electronic form is mandatory. (2) Data selection is

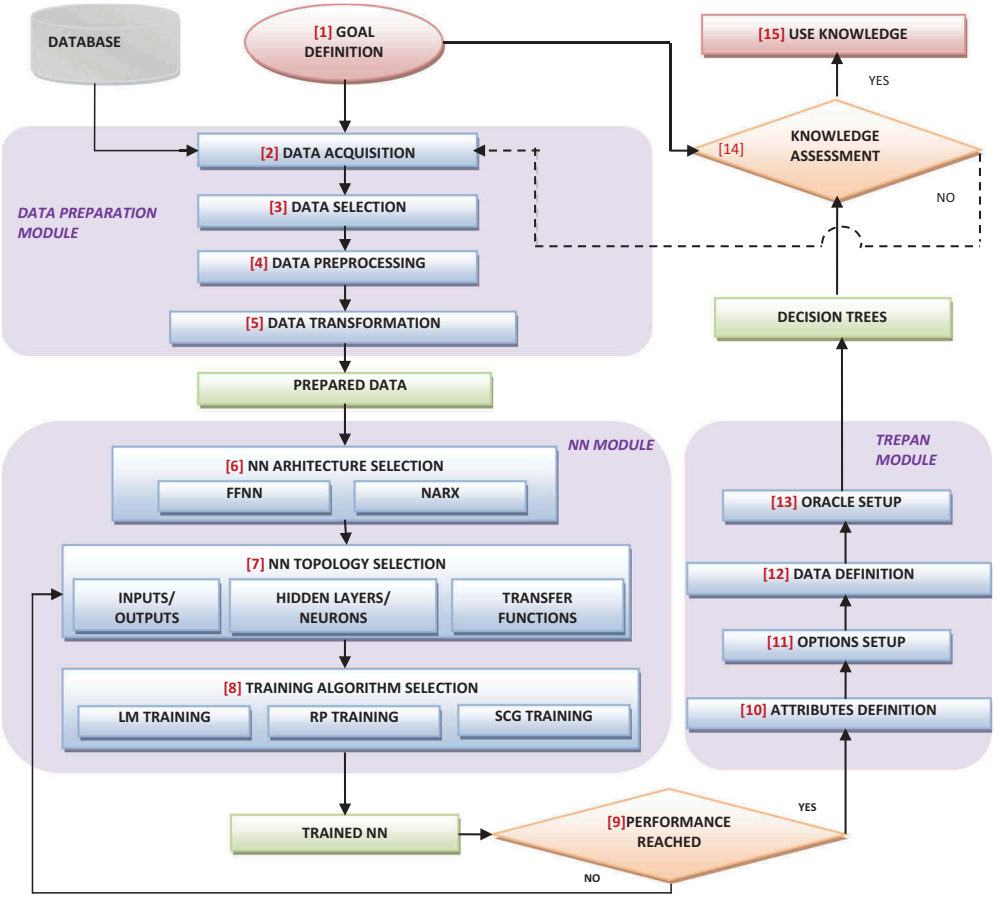

Figure 1. IKDS steps.

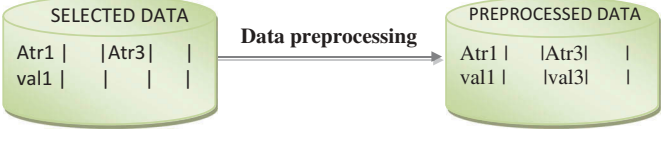

Figure 2. Data preprocessing.

important as all the data samples acquired are relevant for analysis or are related to IM. This phase involves database queries and sampling techniques aiming at the extraction of relevant attributes (variables) from database. The data selection step eliminates unwanted or parasitic data from the sample analysed. For example, a database contains four attributes (Atr1, Atr2, Atr3 and AtrN), each having a value val1, val2, val3 and valN, respectively. After data selection, only Atr1 and Atr3 are kept for the next step. (3) Data preprocessing deals with data cleaning or enrichment. Basic operations involve identifying corrupted or missing data. Even though NN is fault tolerant, it is desirable to recover the missing data. If possible, the performance of the network is enhanced. For example in Figure 2, the value of Atr3 is missing. After the preprocessing step, Atr3 is filled with val3. One method to fill in the missing values of an attribute is by averaging the previous and the next neighbour values, if they exist. For example, if Atr3 has m values and the value kth is missing, then $val_k = \frac{val_{k-1}+val_{k+1}}{2}$.

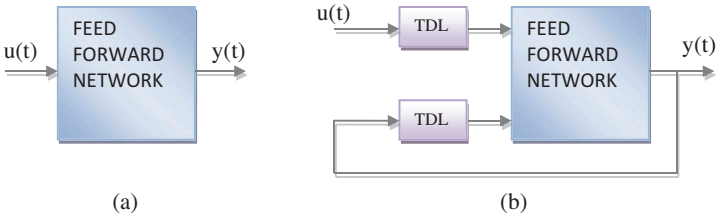

Figure 3. FNN (a) and NARX (b).

Data transformation changes data into a form suitable for input into NN. Not all the data available in IM are suitable to feed the inputs of a NN. For example, the data transformation step can also improve the convergence of the learning methods by mapping the minimum and maximum values of a row to [−1 1] as ANN learns more efficiently if the input variables are small in dimension. Atr1 has a categorical value F (False) while Atr3 has T (True) value. Transforming the data, F becomes 0 while T becomes 1. Now the data is ready for the next step, which is NN architecture selection.

3.2. *Neural network module*

In this research, two ANN architectures are used: feedforward neural network (FNN) in Figure 3(a) and NARX in Figure 3(b). The selection of the architecture to use and train depends on the type of data available. If we are dealing with time series data, a NARX network is more suitable to fit the data. If the database contains relational data, FNN is preferred. Both architectures can be used interchangeably but with reduced performance. The equation for the FNN model is defined as follows:

$y(t) = f(u(t))$, where $u(t)$ is the input and $y(t)$ is the output of the network.

The defining equation for the NARX model is

$$y(t) = f\big(y(t-1), y(t-2), \ldots, y\big(t-n_y\big), u(t-1), u(t-2), \ldots, u(t-n_u)\big)$$

where the next value of the dependent output signal $y(t)$ is regressed on the previous values of the output signal and previous values of an independent (exogenous) input signal. In fact, a NARX network is an FNN plus one or more time delay lines (TDL) at the input and recurrent line.

The scaled conjugate gradient (SCG) (Riedmiller and Braun) is used for both function approximation and pattern recognition tasks, and in topologies with a high number of weights. The SGC algorithm combines the conjugated gradient algorithm with Levenberg–Marquardt algorithm which is one of the most effective second-order methods for searching in multidimensional nonlinear surfaces (Dos Santos, Eurípedes and Von Zuben 2001). The findings from Møller (1993) showed that SCG algorithm has the best performance in the standard backpropagation method than another conjugate gradient algorithm. According to the studies of Zhou and Zhu (2013), SCG can reach convergence with less iteration compared with standard backpropagation algorithms. The algorithm is based upon a class of optimization techniques in numerical analysis known as the conjugate gradient methods. SCG uses the Hessian matrix of the network but requires only X (N) memory usage, where N is the number of weights in the network. A pseudocode to the SCG gradient is described in Figure 4.

The parameters used in training the NN using the SCG algorithm are described in Table 1.

After the ANN is trained, the next step is to verify the performance accuracy of the network. The number of neurons in the hidden layer is adjusted progressively until the performance is satisfactory. On the contrary, if the training performance is satisfied for the function approximation or pattern recognition problem, the next step is to extract knowledge learned by the network.

1. Select weight vector w_1 and scalars $\sigma>0$, $\lambda_1>0$ and $\bar{\lambda}_1=0$.
   Set $p_1= r_1= -E'(w_1)$, k=1 and success=true. {E(w) global error function }
2. **IF** success=true then calculate second order information:
   $\sigma_k= \sigma/|p_k|$ { p_k is non-zero weight vector from training set }
   $s_k=\frac{E'(w_k+\sigma_k p_k)-E'(w_k)}{\sigma_k}$ { s_k is the approximation of the Hessian matrix $E''(w_k)$}
   $\delta_k = p_k^T s_k$ { δ_k is the error }
3. Scale s_k:
   $s_k = s_k + (\lambda_k - \bar{\lambda}_k)p_k$ { λ regulates the indefiniteness of the Hessian matrix}
   $\delta_k = \delta_k + (\lambda_k - \bar{\lambda}_k)|p_k|^2$
4. **IF** $\delta_k<=0$ **THEN** make the Hessian matrix positive define:
   $s_k = s_k + (\lambda_k - 2\frac{\delta_k}{|p_k|^2})p_k$
   $\bar{\lambda}_k = 2(\lambda_k - \frac{\delta_k}{|p_k|^2})$
   $\delta_k = -\delta_k + (\lambda_k)|p_k|^2$, $\bar{\lambda}_k=\lambda_k$
5. Calculate step size:
   $\mu_k = p_k^T r_k$, $\alpha_k = \frac{\mu_k}{\delta_k}$
6. Calculate the comparison parameter: $\Delta_k=\frac{2\delta_k[E(w_k)-E(w_k+\sigma_k p_k)]}{\mu_k^2}$
7. **IF** $\Delta_k>=0$ **THEN** a successful reduction in error can be made:
   $w_{k+1}=w_k + \sigma_k p_k$
   $r_{k+1} = -E'(w_{k+1})$
   $\bar{\lambda}_k=0$, success= true
   7a. **IF** k mod N=0 **THEN** restart algorithm: $p_{k+1}=r_{k+1}$
   ELSE create new conjugate direction:
   $\beta_k=\frac{|r_{k+1}|^2-r_{k+1}r_{k+1}}{\mu_k}$
   $p_{k+1} = r_{k+1} + \beta_k p_k$
   7b. **IF** $\Delta_k>=0.75$ **THEN** reduce the scale parameter: $\lambda_k=\frac{1}{2}\lambda_{k+1}$
   ELSE a reduction in error is not possible: $\bar{\lambda}_k = \lambda_k$, success = false
8. **IF** $\Delta_k<0.25$ then increase the scale parameter: $\lambda_k=4\lambda_k$
9. **IF** the steepest descent direction $r_k \neq 0$ **THEN** set k=k+1 and go to step 2
   ELSE terminate and return w_{k+1} as the desired minimum.

Figure 4. Pseudocode for SCG algorithm.

Table 1. Scaled conjugate gradient training parameters.

Parameter	Description
epochs	The training stops if the number of iterations exceeds the *epoch's* parameter.
goal	Training stops if the performance function drops below the *goal* parameter.
min_grad	Training stops if the magnitude of the gradient is less than the *min_grad* parameter.
max_fail	Used for early stopping.
sigma	Determines the change in the weight for the second derivative approximation.
lambda	Regulates the indefiniteness of the Hessian matrix.
time	Training stops if the training time is longer than the *time* parameter (s).

3.3. *TREPAN module*

The algorithm adopted in this work and used for extracting the knowledge embedded in trained NNs is TREPAN (Craven 1996). TREPAN is adopted for rule extraction and knowledge discovery as it takes the advantages of generality that does not require a defined network architecture or training method (Apté and Weiss 1997). As TREPAN is a best-first search approach, it provides high fidelity of the extracted knowledge from the trained network. Also, it has higher predictive accuracy and it is still being applied in knowledge extraction from the coastal big data (Millie et al. 2013), bioinformatics and cheminformatics. This paper applies TREPAN in a logistics application. The algorithm builds a decision tree representing the function that an NN has learnt. TREPAN is designed to extract a decision tree that maximizes the trade-off between fidelity and concision.

First, it uses the NN's training set to construct models of the data distribution in the problem domain and uses these models to generate instances or membership queries. Second, the algorithm develops trees in a best-first manner, attempting to maximize the gain in fidelity each time it expands a node in the tree.

The function which models the trained NN (named Oracle) coded in Matlab environment is composed of a weights matrix, biases matrix and the activation functions of the hidden and output layers. The algorithm also requires the input/output data which were used in training the NN. The decision tree is generated by the TREPAN algorithm and the node of decision tree stores the indices of its parent and children. By using binary splitting testing, the node has either two or no children. The node with children is represented by a fork while the node without children is represented as a leaf. A suitable class is assigned based on the training result.

TREPAN has been recognized as an important algorithm which can extract rules and represent knowledge learned from the NN. TREPAN is a best-first method by assigning the class as a node with high priority, and subsequently classifying all examples to nodes and expanding the tree. The stopping criteria can be set by the tree size or by deciding whether a leaf should be made for the decision tree or not. Compared with other methods such as Rule Extraction from Ensemble Methods via Tree Induction and Combination (REEMTIC) and ensemble, the accuracy percentage over the problem size of TREPAN is not compatible with REEMTIC and the ensemble classifier. However, TREPAN can extract more rules (Al Iqbal 2012) which is important for knowledge discovery.

Differing from those trees using a breath-first and depth-first search, TREPAN makes use of best-first method. A priority is assigned for each node, and nodes with higher priority are processed first so as to have a more accurate classification of the training data. The new nodes with lower priority are placed in the queue. The tree and the queue are initialized with the root node until all the examples pass through. A splitting test is performed by starting with a set of candidate tests and a binary test is carried out on each value for nominal features, and a binary test on the threshold for real-valued features. Adopting *m*-of-*n* test, the candidate makes use of an operator, so the *M*-of-*N* test (*M*, *N*, *S*) and a split *s* are combined to form an *M*-of-(*N* + 1) test (*M*, *N* + 1, *S* + *s*) and an (*M* + 1)-of-(*N* + 1) test (*M* + 1, *N* + 1, *S* + *s*). TREPAN makes use of a beam search in maximizing the information gain. Additional input patterns are generated by sampling the distributions of the original training. The output value can be assigned by using oracle query for the new training case.

The stopping criteria can be set by fulfilling one of the following conditions: (1) the number of nodes exceeds the predefined maximum size or (2) the queue is empty, where the *min_sample* parameter represents the minimum number of examples that are generated to reach each node. TREPAN ensures that it has at least min sample instances at a node before giving a class label to the node or choosing a splitting test for it. *Query_instances is* a question to an oracle that consists of an instance from the learner's instance space. Given a membership query, the oracle returns the class label for the instance.

TREPAN is not limited to using only the network's training data. However, it makes membership queries for other instances as well. TREPAN calls the DRAWSAMPLE routine to get a set of query instances to use in membership queries. The CONSTRUCTTEST function is used for a splitting test for a node. TREPAN uses *m*-of-*n* expressions for its tests. An *m*-of-*n* expression is a Boolean expression that is specified by an integer threshold, *m*, and a set of *n* Boolean literals. An *m*-of-*n* expression is satisfied when at least *m* of its *n* literals are satisfied. For example, suppose there are three Boolean features, a1, a2 and a3. Two of a1, a2 and a3 are logically equivalent to (a1 and a2), (a1 and a3) or (a2 and a3). During classification, when an example reaches a fork in the tree, it passes down the first branch if the *M*-of-*N* test at that node is satisfied, and down the second branch otherwise. The choice of which *M*-of-*N* test to use at a node is based on maximizing the information gain of the test. To avoid the combinatorial explosion involved in searching the space for all possible *M*-of-*N* tests, a beam search is used. The beam is initialized with the split having the largest information gain and its complement. The beam search proceeds iteratively, constructing increasingly complex *M*-of-*N* tests for possible inclusion in the beam, terminating when the beam remains unchanged during a single iteration. To avoid overfitting the training data, tests are

considered for inclusion in the beam only if the partitioning of the training set produced is significantly different (on the basis of a chi-squared test) from that produced by the original test. A test that satisfies this condition replaces a test in the current beam if it has a larger information gain.

A usual limitation of decision tree methods is that as the tree grows, the number of training cases reaching a node decreases, or there are inadequate examples for expanding the tree further. The most notable aspect of TREPAN aims to overcome the above problem. Additional input patterns are generated by sampling the distributions of the original training data, and the corresponding output is produced by applying the oracle. Discrete variables are modelled by their empirical distributions, and kernel density estimates are used to model continuous inputs. As each node is created, sufficient training cases are generated so that a prespecified minimum number of examples reach the node. The generation of new input patterns is based only on the marginal distributions of the features and ignores any correlations between the features. To partially address this deficiency, the tool estimates the local distributions of the original training cases reaching each node. If these distributions are significantly different (using a chi-squared test for discrete variables and a Kolmogorov–Smirnov test for continuous variables) from the distributions at the parent node, then the local distributions are used to generate new input patterns. Otherwise, the parent distributions are used.

4. Evaluating of IKDS

This section describes the solution to an inventory problem encountered by a company, aliased as PRD Pte Ltd, located in Singapore, using the framework proposed in Section 3. The problem encountered by PRD Company was the necessity of having a suitable forecasting technique used for short periods of time, more exactly 2 weeks. Even though the sales department provides the information necessary to produce new compressors, it seems that this information is not stable over time, varying from 90% accuracy for a short time span to 60% for a mid time span and about 30% for long time span. As the company lacks a suitable forecasting technique for short periods of time, a NARX network is proposed to forecast the demand. The TREPAN algorithm is used to extract a decision tree which could be used by managers to improve the inventory level. The IKDS is tested in two parts: (1) a time series forecasting of the demand in which the data is recorded over a period of 3 years and represents the demand of one item carried by the case company and (2) an analysis of the impact of demand, lead time and cycle service level (CSL) on SS. The related inventory data is used for illustration.

4.1. *Time series forecasting of the demand*

The first case is a time series forecasting problem. Table 2 shows that data used were recorded over a period of 3 years and represent the demand of one item carried by the company.

The purpose is to train a NARX network to forecast demand on month 36 (Figure 5). Therefore, the first 35 values are used for training the network.

Training of the NARX network is done by using the first three values of the time series data as the inputs and the fourth as the target. Considering time series data t_1, t_2, t_3, t_4, …, t_n with n values (in our case $n = 35$ months), the first three values t_1, t_2 and t_3 are used for training and the next value t_4 is used as the target. For the second training set, t_2, t_3 and t_4 are used for training and t_5 is used as the target. The procedure is repeated for $n - 3$ months. Therefore, the NARX network has

Table 2. Demand over 36 months.

Month	1	2	3	4	5	6	7	8	9	10	11	12
Year 1	64	59	65	73	74	86	68	40	35	66	97	64
Year 2	75	54	25	70	48	68	64	35	35	26	51	51
Year 3	27	48	25	60	26	41	32	37	57	23	39	21

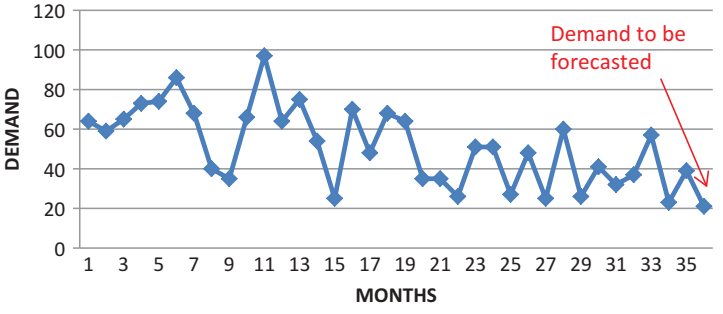

Figure 5. Distribution of demand over time.

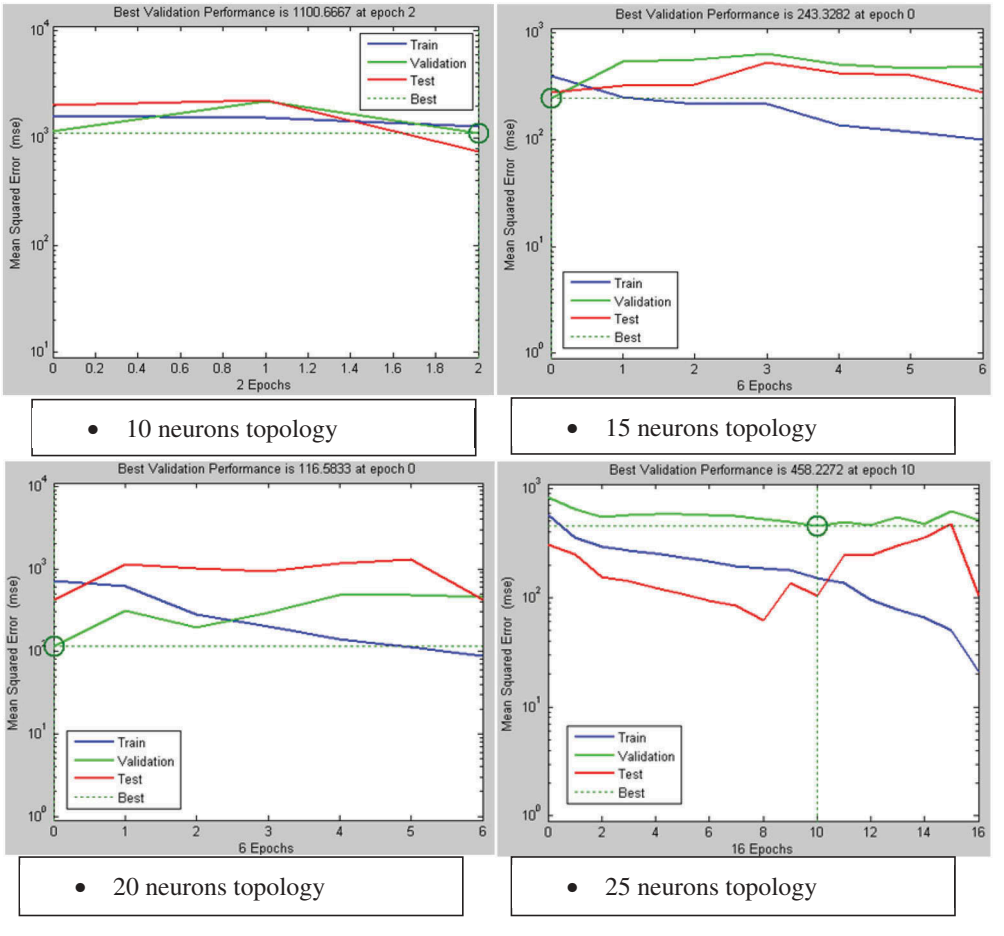

Figure 6. Performance assessment for four NARX topologies (best validation performance obtained for 20 hidden neurons topology).

four inputs (considering the recurrent input also) and one output. A sample of the data used for training is depicted. Four NARX networks are trained using 10, 15, 20 and 25 neurons in hidden layer. In Figure 6, it can be observed that the best training performance is obtained for a 20 neurons topology. After the network is trained, the next step is to forecast the demand for the 36th

Table 3. Backpropagation training parameters for the causal forecasting case.

max_fail	Parameter	epochs	show	goal	time (s)	min_grad	delt_inc	delt_dec	delta0	deltamax
6	Value	1000	5	0	60	1e-010	1.2	0.5	0.07	50

Table 4. Forecasted values.

	NARX	ARIMA(1,1,1)	MA(3)	MA(2)	Actual value
Forecasting	23	30.11	39.66	26	21
Error	2	9.11	18.66	5	0

NARX: autoregressive exogenous neural network; ARIMA: autoregressive integrated moving average; MA(3): third-order moving average; MA(2): second-order moving average.

month. Using the previous three values from the time series (57, 23 and 39) as input, the output of the network is 23. The actual value of the time series for the 36th month is 21. Therefore, the error is $e = 23–21 = 2$. The parameters used in training the network are depicted in Table 3.

The training will stop if the epochs reach 1000, the time reaches 60 s, the max_fail reaches 6 or the goal is achieved by minimizing the performance function (mean square error) to zero.

In Table 4, the NARX network accuracy is compared with the ARIMA, the second-order moving average MA(2) and the third-order moving average model MA(3). The MA has been widely used by researchers with different time series methods such as exponentially weighted MA, multifractal MA and ARMA. ARIMA is derived from the fundamental concept of the MA. Therefore, MA(2) and MA(3) are used for comparative analysis. A hybrid intelligent system combining ARIMA models and ANN is proposed for demand forecasting to obtain a lower inventory level in a practical case. Jaipuria and Mahapatra (2014) compared their proposed discrete wavelet transformation-ANN approach with ARIMA. The modified regression model is used to predict the demand of service parts. Since most researchers have derived a new model with ARIMA or done comparisons with ARIMA, this paper also conducts comparison analysis with ANN and ARIMA, as shown in Table 4. It can be observed that the best accuracy in forecasting demand for the 36th month is obtained by the NARX network.

In order to use TREPAN and extract the knowledge learnt by the NARX network, some modifications of the network and data have to be done. The slopes between time series values are calculated and encoded. The encoding is made in order to mediate the comprehensibility of the rules extracted by TREPAN algorithm.

The slope (trend) is one of the most important features in the time series and is calculated in Last, Klein, and Kandel (2001) as

$$\propto_i = \frac{y(t_{i+1}) - y(t_i)}{t_{i+1} - t_i} \tag{1}$$

Next, the slopes are coded as in Figure 8. **IF** $\propto_i \geq 0$ **THEN** $alpha_i = 1$, **ELSE** $alpha_i = 2$.

If the slope is positive or equal to zero, then encoded as '1'. If slope is negative, then coded as'2'. $Alpha_i$ denotes the coded value of the slope $\propto_i$ (Figure 7).

The idea is to use the first three values (t_1, t_2 and t_3) and their slopes (alpha1, alpha2 and alpha3) to predict if the fourth slope is positive or equal to zero (coded as 1) or is negative (coded as 2). Note that alpha2 is the slope calculated between the values t_2 and t_3, while alpha3 is the slope calculated between t_1 and t_3 (Figure 8).

The variables used to feed the NARX network are depicted in Table 5. A sample used to train the NN is depicted.

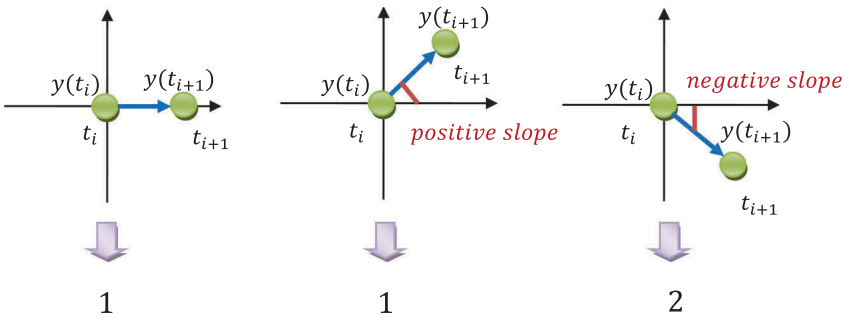

Figure 7. Proposed coding of slopes for time series case.

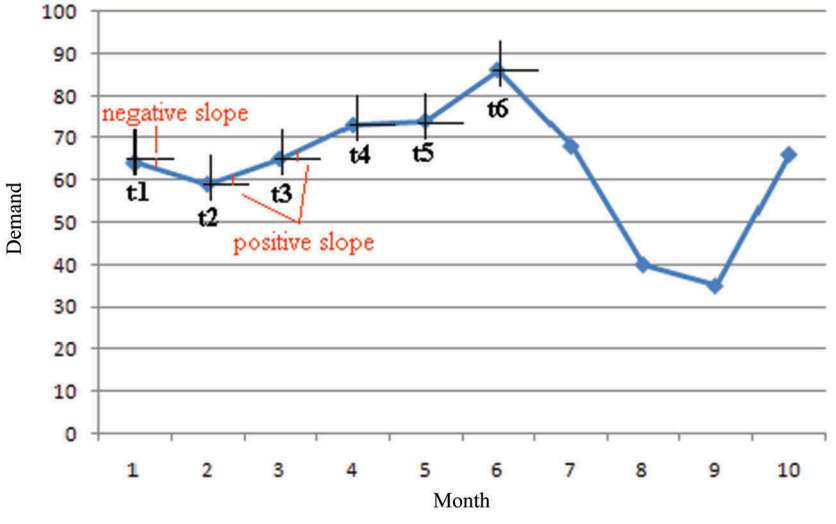

Figure 8. Slopes representation for each value of demand.

Table 5. Variable selection for time series case.

Variables	Abbreviation	Type	Range of values
Level t_1	l1	R (real)	21–97
Slope 1	alpha1	N (nominal)	1,2
Level t_2	l2	R	21–97
Slope 2	alpha2	N	1,2
Level t_3	l3	R	21–97
Slope 3	alpha3	N	1,2
Class		N	Down/up

The tree extracted using TREPAN is shown in Figure 10, and node 1 of Figure 11 is used for illustration.

If both of the following conditions are satisfied,

(1) alpha3 = 1 (i.e. slope 3 is positive or zero) and
(2) *l*3 > 56 (i.e. demand a level t_3 shown in Figure 9 is more than 56),

then it is predicted that the demand will increase.

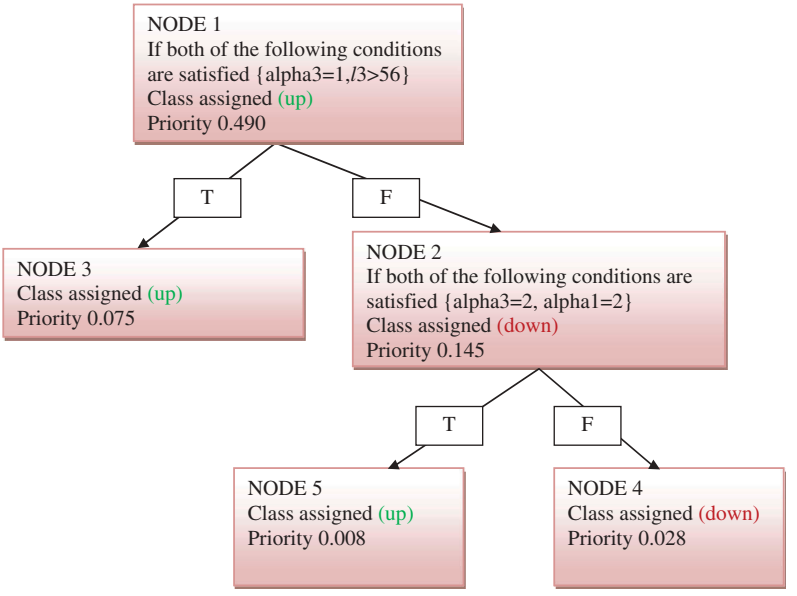

Figure 9. Tree extracted by TREPAN for time series case.

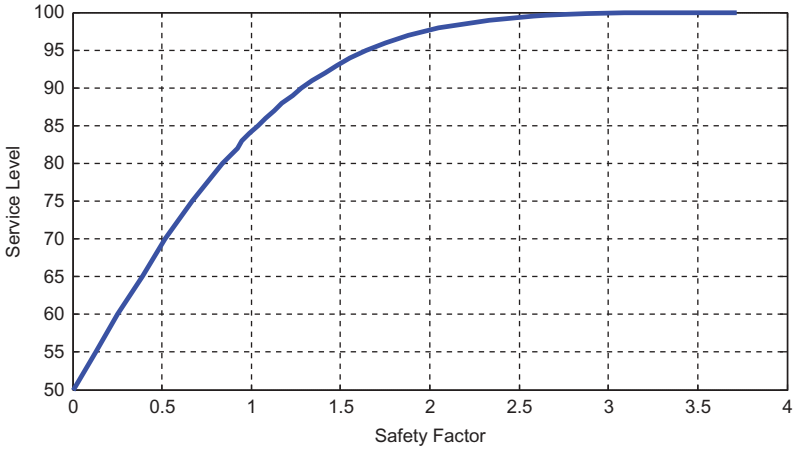

Figure 10. Relation between service level and safety factor.

If the case does not have same condition as node 1, then go to node 2. In node 2, if both slope 3 and slope 1 decrease, the demand will decrease.

4.2. *Impact of demand, lead time and CSL on safety inventory*

For the second experiment, the inventory levels are analysed with the purpose of determining the impact of demand, lead time and CSL on SS. The idea is to determine if safety stock SS_i at time period t_i is going to decrease or increase with respect to the previous safety stock (SS_{i-1} at time period t_i).

Recall that SS is the inventory carried to improve product availability in the presence of uncertainty, in both demand and supply. Therefore, the appropriate level of SS is calculated with respect to the uncertainty of demand and supply, and the desired level of product availability.

If the uncertainty of demand and supply increases, the required level of SS also increases. If the preferred level of product availability increases, the required level of SS also increases (Figure 10).

Consider the situation of a continuous review system (Figure 12). The level of SS at time period t_i is calculated as $SS_i = norminv(CSL)\sigma_{dl_i}$, where *norminv()* is the normal inverse cumulative distribution function, the CSL has a range between 0.5 and 0.99, and σ_{dl_i} is standard deviation of demand during the lead time. The formulas to calculate the variables involved in calculating SS at time t_i are given below:

$$\text{Standard deviation of demand during lead time } \sigma_{dl_i} = \left(L_i{\sigma_{d_i}}^2 + {D_i}^2{\sigma_{l_i}}^2\right)^{\frac{1}{2}} \tag{2}$$

$$\text{Average demand, } D_i = \frac{1}{N}\sum_{i=1}^{N} d_i \tag{3}$$

$$\text{Standard deviation of demand per period, } \sigma_{d_i} = \left(\frac{1}{N-1}\sum_{i=1}^{N}\left(d_i - \frac{1}{N}\sum_{i=1}^{N} d_i\right)^2\right)^{\frac{1}{2}} \tag{4}$$

$$\text{Average lead time for replenishment, } L_i = \frac{1}{N}\sum_{i=1}^{N} l_i \tag{5}$$

$$\text{Standard deviation of lead time, } \sigma_{l_i} = \left(\frac{1}{N-1}\sum_{i=1}^{N}\left(l_i - \frac{1}{N}\sum_{i=1}^{N} l_i\right)^2\right)^{\frac{1}{2}} \tag{6}$$

Figure 11 shows the continuous review inventory model. Inventory status is continuously checked and an order is placed when the inventory drops below the reorder point, and T_i is the time between successive orders.

Demand on time period t for PCs at Dell is normally distributed and using the above equations, the variables used to calculate the SS level are depicted in Table 6.

Variables used to train the TREPAN algorithm are given in Table 7.

The numbers are generated randomly with $i = 250$ samples of data. As ANN cannot be trained using random inputs/outputs values, a mapping function must be used to correlate the inputs with the outputs:

```
if safety_stock(i)≥safety_stock(i-1)
class(i) = 1; %increase
else class(i) = −1; %decrease
end;
```

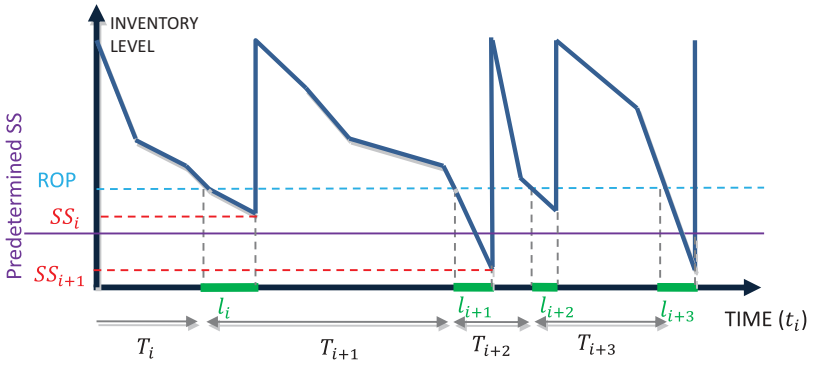

Figure 11. Continuous review.

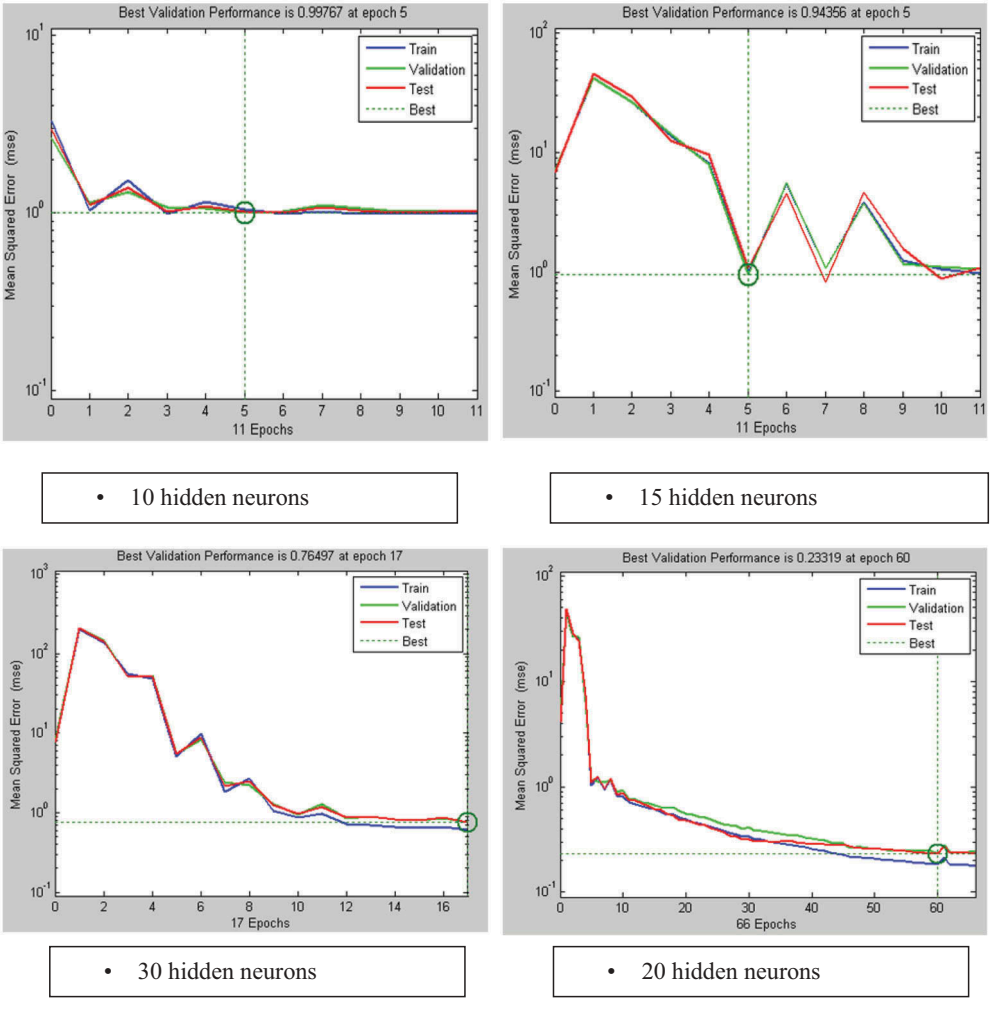

Figure 12. Training performance achieved by tested network topologies (best performance achieved by 20 hidden neurons topology).

In other words, IF the SS at time period t_i is greater or equal to the previous SS (period t_{i-1}), then the class assigned is 'increase' (the SS has increased) ELSE class assigned is 'decrease' (the SS has decreased). The coding in Matlab generates the numbers, and the network is formed and trained.

This is a pattern recognition problem with an increased number of inputs and weights, therefore, the training algorithm is the SCG and an FNN is selected as the architecture. From 250 samples, 60% are used for training, 20% for validation and last 20% for testing. After training four separate FNN networks with 15, 20, 25 and 30 neurons in the hidden layer, the topology with 20 neurons obtained the best results in fitting the data. The best validation performance is at 0.233 for epoch 60 (Figure 12).

The top left of Figure 12 depicts the training performance of the training set (60% from the total 250 samples), the top right of Figure 12 depicts the performance of the validation set (20%) and the bottom left of Figure 12 shows the performance of the test set (20%).Training stops when the validation set is optimal (goal is 0).

Table 6. Sample from set used to train the NN.

Period (t_i)	1	2	3	4	5	6	7	8	9	10
Demand (d_i)	2159	4045	8762	3653	9351	3213	5462	2923	4508	7236
Lead time (l_i)	8	12	3	14	4	8	3	3	14	9
Average demand		3102	4989	4655	5594	5197	5235	4946	4897	5131
Standard deviation demand		1334.069	3401.22	2856.28	3244.97	3060.85	2795.95	2714.5	2543.38	2509.4
Average lead time		10	7.66667	9.25	8.2	8.16667	7.42857	6.875	7.66667	7.8
Standard deviation lead time		2.828427	4.50925	4.85627	4.81664	4.30891	4.39155	4.3569	4.71699	4.4672
Safety stock		0	3959	9578	5333	5981	1737	44,913	9947	24,660
Cycle service level		0.657879	0.65275	0.58715	0.5831	0.5288	0.96816	0.6694	0.84639	0.9664

Table 7. Variable selection for TREPAN training.

Variables	Type	Range of Values
mean_demand	R (Real)	
std_dev_demand	R	
mean_lead_time	R	
std_dev_lead_time	R	
previous_safety_stock	R	
cycle_service_level	R	
class	N	Increase/decrease (1 or −1)

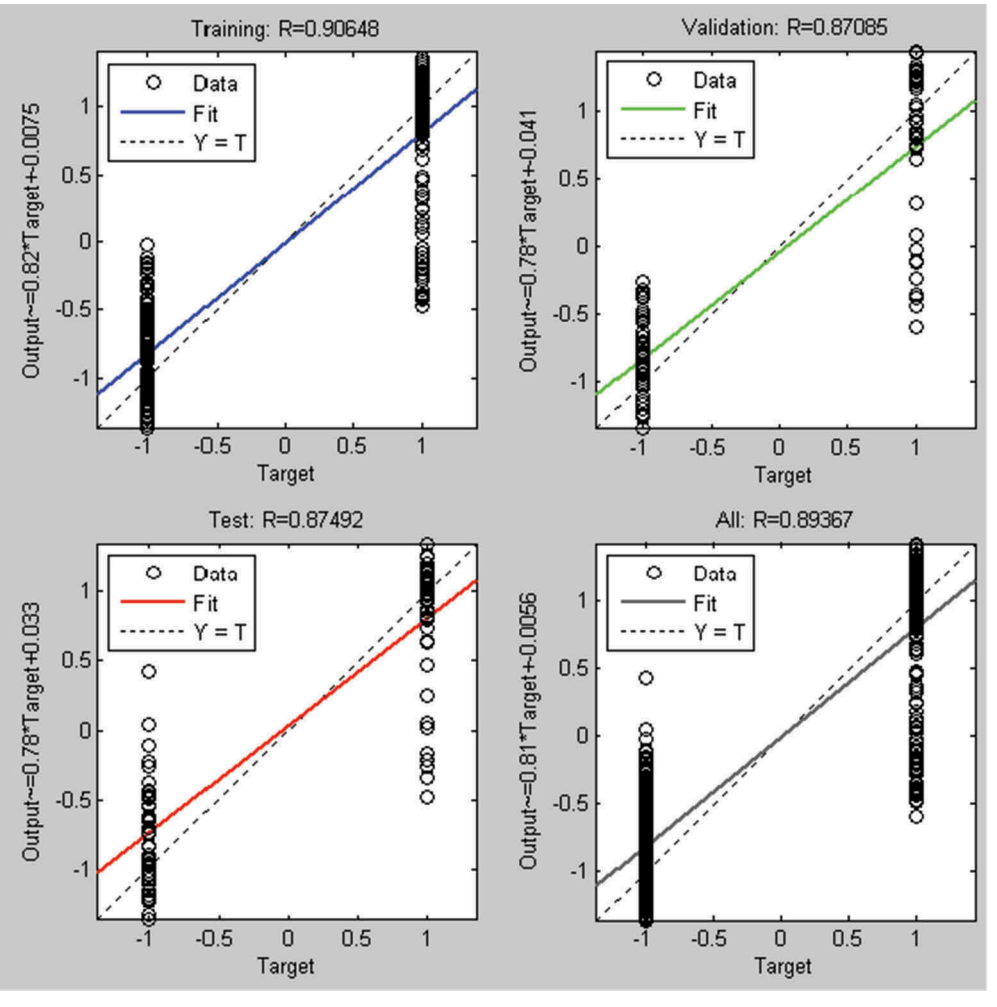

Figure 13. Regression analysis for 20 neurons topology selected.

Regression analysis (Figure 13) is a measure of how well the network fits the input vector variables to the output vector variables. In an ideal case, *R* (regression gradient) is 1. The NN selected in this simulation (20 neurons topology) performs satisfactorily with values around 0.9 for training sample, 0.87 for validation set and 0.874 for test samples.

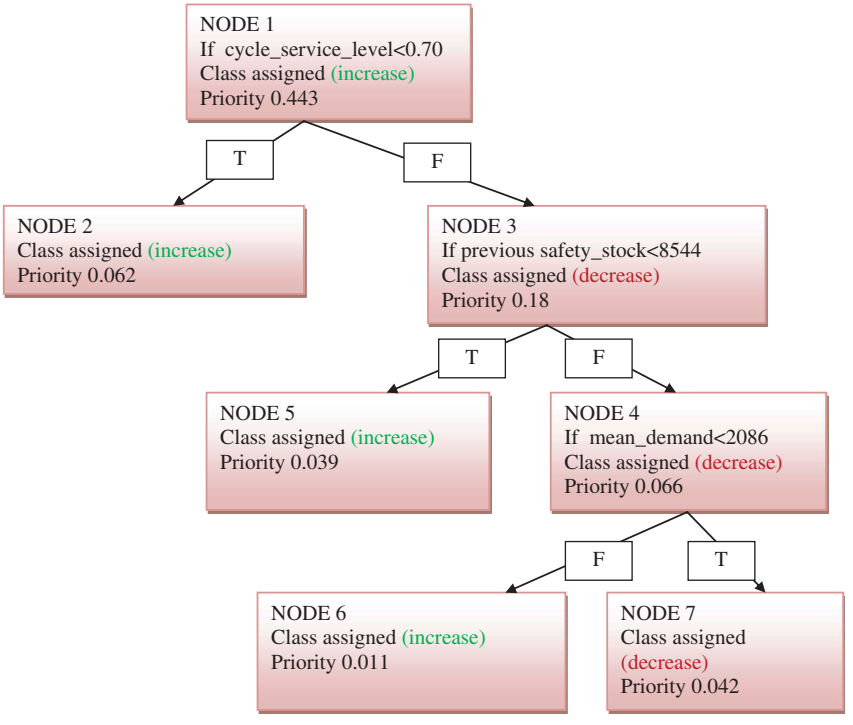

Figure 14. Tree extracted by TREPAN.

After training the network, the TREPAN algorithm is used to extract learnt relationships by the network. Input data are used to train the ANN. An interpretation of the extracted decision tree is shown in Figure 14.

4.3. *Discussion and conclusions*

The efficiency of the IKDS has been verified in two cases: The first analysis is a time series forecasting of demand of one item carried by the case company and the second case is the analysis of the impact of demand, lead time and CSL on the SS.

In the first part of the case study related to time series forecasting, it can be concluded that the NARX network has a superior performance in comparison with traditional forecasting methods (ARIMA or MA). The TREPAN algorithm can be used to extract knowledge embedded in the NARX network to explain the functionality of the network. The interpretation of the tree extracted in the causal forecasting case can be done as follows: IF both slope 3 is positive and l3 (level t_3) is bigger than 56, THEN next slope is positive (up). If only one of them is true or both are false, then the next slope is negative (down). For node 2, if both slope 3 and slope 1 are negative, then the next slope is positive, else the slope is negative. The rules have high fidelity with almost all the time series data used in the training process. An inventory manager could use this knowledge in the form of rules to manage inventory more efficiently.

The second part of the case study has the purpose of understanding the available data, more exactly to determine the impact of demand, lead time and CSL on SS calculation at the case company. Therefore, an FNN is trained with the purpose to learn the relations between the input variables (average demand, standard deviation of demand, average lead time, standard deviation of lead time, previous SS and CSL) and output which is '1' if the calculated SS at a period t of time has increased in comparison to the previous SS (at time $t - 1$), or '–1' if it has decrease. Interpreting the decision tree

obtained using the TREPAN algorithm can be done as follows: For node 1, IF the cycle_service_level is less than 0.70, then the next SS level will decrease in comparison to the actual one (node 2), ELSE for all cycle_service_level ≥ 0.7, IF the previous_safety_stock (node 3) is less than 8544 units, then the next SS will increase also, ELSE for all cycle_service_level ≥ 0.7 and previous_safety_stock ≥ 8544, IF mean_demand <2086. then the next SS will increase, ELSE next SS will decrease.

5. Conclusions

This work has made valuable contributions in the application of advanced data analysis techniques to IM. An IKDS was developed with the purpose of performing two tasks: IKDS helps to (1) forecast demand in IM and (2) get a better understanding of the variables involved in the forecasting process. The first task is achieved by using an ANN while the second task is achieved by using the TREPAN algorithm which extracts the knowledge learnt by a trained NN in the form of decision trees. Extracting the knowledge embedded in ANN, weights could help in understanding the relevance of the input parameters and the factors which are detrimental to IM. Decision rules embodied in the decision tree generated by TREPAN algorithm help users in decision-making.

The performance of the proposed system (IKDS) has been carefully evaluated and verified in two situations. One is a time series forecasting problem for demand and another is related to lead time and CSL on warehouse SS. A real industrial application in a case company has been included to illustrate the efficiency of the proposed system in industry.

The proposed methodology is a step towards the development of IKDSs for extracting the knowledge and hidden patterns from inventory data. The results show that forecasting with ANN promises superior accuracy in comparison to traditional methods like MA or ARIMA. Apart from adopting NNs to increase the accuracy of the forecasting result, this report has contributed to providing explanations of models and patterns learnt by the ANN with the TREPAN algorithm. Instead of determining the exact solution of the inventory level and inventory position, this study pioneers the adoption of knowledge discovery techniques to analyse the reasons behind the inventory level trends. Through studying the slope of the inventory trend, industrial practitioner can make a decision to adjust the inventory and extract knowledge in terms of rules, so as to be responsive to the dynamic business environment.

Acknowledgements

Our gratitude is extended to the research committee and the Department of Industrial and Systems Engineering of the Hong Kong Polytechnic University and the Nanyang Technological University for support in this project.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

Our gratitude is extended to the research committee and the Department of Industrial and Systems Engineering of the Hong Kong Polytechnic University and the Nanyang Technological University for support in this project.

References

Abdel-Aal, R. E. 2008. "Univariate Modeling and Forecasting of Monthly Energy Demand Time Series Using Abductive and Neural Networks." *Computers & Industrial Engineering* 54 (4): 903–917. doi:10.1016/j.cie.2007.10.020.

Al Iqbal, M. R. 2012. "Rule Extraction from Ensemble Methods Using Aggregated Decision Trees." In *Neural Information Processing*. Springer, Berlin Heidelberg.

Anderson, J. A., and J. Davis. 1995. *An Introduction to Neural Networks*. 1 Vol. Cambridge, MA: MIT Press.

Augasta, M. G., and T. Kathirvalavakumar. 2012. "Rule Extraction from Neural Networks—A Comparative Study." In *Pattern Recognition, Informatics and Medical Engineering (PRIME), 2012 International Conference on IEEE,* Salem, Tamilnadu, March, 404–408.

Bowersox, D. J., D. J. Closs, and M. B. Cooper. 2002. *Supply Chain Logistics Management*. Vol. 2. New York: McGraw-Hill.

Box, G. E., G. M. Jenkins, and G. C. Reinsel. 2016. *Time Series Analysis: Forecasting and Control*. 734 Vols. Hoboken, NJ: John Wiley& Sons.

Carbonneau, R., K. Laframboise, and R. Vahidov. 2008. "Application of Machine Learning Techniques for Supply Chain Demand Forecasting." *European Journal of Operational Research* 184 (3): 1140–1154. doi:10.1016/j.ejor.2006.12.004.

Carmo, J. L., and A. J. Rodrigues. 2004. "Adaptive Forecasting of Irregular Demand Processes." *Engineering Applications of Artificial Intelligence* 17 (2): 137–143. doi:10.1016/j.engappai.2004.01.001.

Chang, P. C., Y. W. Wang, and C. Y. Tsai. 2005. "Evolving Neural Network for Printed Circuit Board Sales Forecasting." *Expert Systems with Applications* 29 (1): 83–92. doi:10.1016/j.eswa.2005.01.012.

Apté, C., and S. Weiss. 1997. Data Mining with Decision Trees and Decision Rules. *Future generation computer systems* 13 (2): 197-210.

Chopra, S., and P. Meindl. 2007. "Supply Chain Management. Strategy, Planning & Operation." Das Summa Summarum des Management, 265–275. Upper Saddle River, NJ: Gabler.

Craven, M. W. 1996. "Extracting comprehensible models from trained neural networks." Doctoral dissertation, University of Wisconsin–Madison.

Craven, M. W., and J. W. Shavlik. 1996. "Extracting Tree-Structured Representations of Trained Networks." In *Advances in Neural Information Processing Systems, Proceedings of the 1995 conference*, edited by D. S. Touretzky, M. C. Mozer and Michael E. Hasselmo, *8 vols.* 24–30. Cambridge, MA: The MIT Press.

Dhar, V. 2013. "Data Science and Prediction." *Communications of the ACM* 56 (12): 64–73. doi:10.1145/2534706.

Dos Santos, Eurípedes, P., and F. J. Von Zuben. 2001. "Efficient Second-Order Learning Algorithms for Discrete-Time Recurrent Neural Networks." *Recurrent Neural Networks: Design and Applications*. 1st Edition. CRC Press.

Elgendy, N., and A. Elragal. 2014. "Big Data Analytics: A Literature Review Paper." Springer's Lecture Notes in Artificial Intelligence (LNAI), 8557. 214–227.Springer International Publishing.

Fu, L. 1991. "Rule Learning by Searching on Adapted Nets." In *Proceedings of the Ninth National Conference on Artificial Intelligence*, 2 vols. 590–595. AAAI Press. July.

Gutierrez, R. S., A. O. Solis, and S. Mukhopadhyay. 2008. "Lumpy Demand Forecasting Using Neural Networks." *International Journal of Production Economics* 111 (2): 409–420. doi:10.1016/j.ijpe.2007.01.007.

Harvey, R. L. 1994. *Neural Network Principles*. Englewood Cliffs, NJ: Prentice-Hall, Inc.

Hruschka, E. R., and N. F. Ebecken. 2006. "Extracting Rules from Multilayer Perceptrons in Classification Problems: A Clustering-Based Approach." *Neurocomputing* 70 (1): 384–397. doi:10.1016/j.neucom.2005.12.127.

Jaipuria, S. 2014. "An Improved Demand Forecasting Method to Reduce Bullwhip Effect in Supply Chains.„ *Expert Systems with Applications* 41 (5): 2395–2408.

Last, M., Y. Klein, and A. Kandel. 2001. "Knowledge Discovery in Time Series Databases." *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)* 31 (1): 160–169. doi:10.1109/3477.907576.

Lee, C. K. M., W. Ho, G. T. Ho, and H. C. Lau. 2011. "Design and Development of Logistics Workflow Systems for Demand Management with RFID." *Expert Systems with Applications* 38 (5): 5428–5437. doi:10.1016/j.eswa.2010.10.012.

Li, L. 2012. "Effects of Enterprise Technology on Supply Chain Collaboration: Analysis of China-Linked Supply Chain." *Enterprise Information Systems* 6 (4): 55–77. doi:10.1080/17517575.2011.639904.

Lieberman, M. B., S. Helper, and L. Demeester. 1999. "The Empirical Determinants of Inventory Levels in High-Volume Manufacturing." *Production and Operations Management* 8 (1): 44–55. doi:10.1111/j.1937-5956.1999.tb00060.x.

Lin, T., B. G. Horne, P. Tino, and C. L. Giles. 1996. "Learning Long-Term Dependencies in NARX Recurrent Neural Networks." *IEEE Transactions on Neural Networks* 7 (6): 1329–1338. doi:10.1109/72.548162.

Menezes, Jr, J. M. P., and G. De A. Barreto. 2006. "A New Look at Nonlinear Time Series Prediction with Narx Recurrent Neural Network." Proceedings of the Ninth Brazilian Symposium on Neural Networks (SBRN'06), October 23-27, 28–33. IEEE. http://ieeexplore.ieee.org/document/4026828/?arnumber=4026828.

Millie, D. F., G. R. Weckman, I. I. Young, W. A. Ivey, J. E. Fries, D. P. Ardjmand, E. Fahnenstiel, and G. L. Coastal. 2013. "'Big Data' and Nature-Inspired Computation: Prediction Potentials, Uncertainties, and Knowledge Derivation of Neural Networks for an Algal Metric.„ *Estuarine, Coastal and Shelf Science*. 125 vols. 57–67. 1 July.

Møller, M. F. 1993. "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning." *Neural Networks* 6 (4): 525–533. doi:10.1016/S0893-6080(05)80056-5.

Predictions, I. D. C.. 2012. Competing for 2020. *2012-10-20]*. http://www. idc. com/getdoc. jsp.

Qi, L. 2013. "A Continuous-Review Inventory Model with Random Disruptions at the Primary Supplier." *European Journal of Operational Research* 225 (1): 59–74. doi:10.1016/j.ejor.2012.09.035.

Russom, P., 2011. *Big Data Analytics*. TDWI Best Practices Report, Fourth Quarter, 1-35. TDWI Research.

Siegelmann, H.T., B. G. Horne, and C. L. Giles. 1997. "Computational Capabilities of Recurrent Narx Neural Networks."." *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 27 (2): 208–215.

Thrun, S. 1993. *Extracting Provably Correct Rules from Artificial Neural Networks*. Sekretariat für Forschungsberichte, Inst. für Informatik III.

Torres, D. E. D., and C. M. S. Rocco. 2005. "Extracting Trees from Trained SVM Models Using a Trepan Based Approach." In *Hybrid Intelligent Systems, 2005. HIS'05. Fifth International Conference on*, November 6-9, 353-360. IEEE. doi:10.1109/ICHIS.2005.41.

Towell, G. G., and J. W. Shavlik. 1993. "Extracting Refined Rules from Knowledge-Based Neural Networks." *Machine Learning* 13 (1): 71–101. doi:10.1007/BF00993103.

Wang, G. C., and C. L. Jain. 2003. *Regression Analysis: Modeling & Forecasting*. Great Neck, NY: Graceway Publishing Company.

Yan, X., Y. Ji, and Y. Wang. 2012. "Supplier Diversification under Random Yield." *International Journal of Production Economics* 139 (1): 302–311. doi:10.1016/j.ijpe.2012.05.013.

Yelland, P. M. 2010. "Bayesian Forecasting of Parts Demand." *International Journal of Forecasting* 26 (2): 374–396. doi:10.1016/j.ijforecast.2009.11.001.

Zhou, F. Y., and X. F. Zhu. 2013. "Alphabet Recognition Based on Scaled Conjugate Gradient BP Algorithm." In *Proceedings of the 2nd International Conference on Green Communications and Networks 2012 (GCN 2012): Volume 1*, edited by Y. Yang and M. Ma. Springer Berlin Heidelberg.