Design and development of a knowledge discovery system in inventory management

Mitrea, Catalin

2010

Mitrea, C. (2010). Design and development of a knowledge discovery system in inventory management. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/38766>

<https://doi.org/10.32657/10356/38766>

**DESIGN AND DEVELOPMENT OF A KNOWLEDGE
DISCOVERY SYSTEM IN INVENTORY
MANAGEMENT**

MITREA CATALIN

School of Mechanical and Aerospace Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Master of Engineering

2010

Abstract

The ability to learn inductively from examples is an important feature of intelligent systems. These learning methods and algorithms, which are able to generate a model of a system using the observations (data) available, are applied in a specific domain usually with the following goals: *to create a system that can carry out a task* and *to obtain a better understanding of the available data*. In this thesis, function approximation or pattern recognition is used to predict inventory level so as to enhance the performance of supply chain.

Inventory management plays a crucial role in logistics and its performance is shaped dramatically by uncertainty in both demand and supply. Although demand is difficult to predict due to its stochastic behavior, it is necessary to have accurate forecasting so as to fulfill the customer needs and maintain the corporate competitive edge. Achieving accurate demand forecasting and representing knowledge in rules attract the attention of both academic and practitioners. Knowledge is regarded as a valuable asset for enterprises and knowledge can be manipulated through Artificial Intelligence (AI) techniques.

Among those AI techniques, Artificial Neural Networks (ANN) has been proven to be very useful in approximating functions and recognizing patterns by adjusting their weights and biases during training process, but it cannot explain how they arrived at a conclusion in solving a problem. This is the reason why ANN is considered in literature as black boxes due to the lack of clear explanation. To overcome these shortages, TREes PARroting Networks (TREPAN) algorithm is used to extract knowledge from the trained networks in the form of decision trees. The TREPAN algorithm utilizes the IF M-of-N THEN conditions rule instead of the common IF-THEN-ELSE rule to construct the decision tree which can be used to understand previously unknown relationship between input variables for forecasting so as to improve inventory management. The experimental results show that forecasting accuracy using ANN are superior to traditional methods like moving average and autoregressive integrated moving average (ARIMA). Also, the knowledge extracted from trained ANNs using TREPAN algorithm is represented in a comprehensible way and can be used to facilitate human decision-making. The significance of this study is to design and develop a knowledge discovery system, thereby enhancing the inventory management.

Acknowledgements

Finishing this work would not be possible without generous help of many people.

First, and foremost, I would like to express my sincere gratitude and appreciation to my supervisor, Dr. Carman Lee for her time, unmeasured patience, inspirations and advices throughout this entire process. She was always there pushing me, challenging me and encouraging me in every possible way.

Another important professor is Dr. Roxana Vasilescu; she was always available to help. Her guidance helped me to shape my research model and enhance my writing skills.

I am thankful to the entire group AIS (Advanced Industrial Systems) for their valuable advices and of which I am proud to be a member.

I would like to thank to technicians from ADAM lab for their support with the software and tools necessary to simulate and test the results.

Table of Content

Abstract.....	I
Acknowledgements	II
Table of Content.....	III
List of Figures.....	VI
List of Tables	VIII
CHAPTER 1. INTRODUCTION	1
1.1 Background.....	1
1.2 Problem statement.....	2
1.3 Motivations	2
1.4 Objectives	3
1.5 Scope.....	4
1.6 Layout of the report	5
CHAPTER 2. LITERATURE REVIEW	6
2.1 Inventory management.....	6
2.2 Demand Forecasting	10
2.2.1 Time Series Models	12
2.2.2 Causal forecasting.....	13
2.3 Knowledge discovery Process	16
2.3.1 Data Mining task and methods	19
2.3.1.1 Clustering.....	19
2.3.1.2 Classification.....	20
2.3.1.3 Prediction	20
2.4 Knowledge Representation	21
2.4.1 Set of rules	21
2.4.2 Decision-Trees	22
2.5 Forecasting with Neural Networks	23
2.5.1 Neural Networks in Logistics	24
2.5.2 Multilayer Feedforward Neural Networks.....	26
2.5.3 NARX Neural Network	28
2.5.4 Techniques used for variable selection.....	30
2.5.4.1 Sensitivity Analysis (I/O selection)	30
2.5.4.2 Committees of Predictors.....	30
2.5.5 Rule extraction from trained neural networks	31

2.6 Implications from the Literature Review	33
CHAPTER 3. DESIGN AND DEVELOPMENT OF A KNOWLEDGE DISCOVERY SYSTEM IN INVENTORY MANAGEMENT	35
3.1 Inventory Knowledge Discovery System (IKDS)	35
3.2 Goal definition (step 1)	37
3.3 Data Preparation module.....	37
3.3.1 Data acquisition (step 2)	37
3.3.2 Data selection (step 3).....	37
3.3.3 Data preprocessing (step 4).....	38
3.3.4 Data transformation (step 5)	38
3.4 Neural Network Module	39
3.4.1 NN Architecture selection (step 6)	39
3.4.2 NN Topology selection (step 7).....	40
3.4.2.2 Number of hidden layers and neurons in the hidden layer	40
3.4.2.3 Activation function selection (Transfer function).....	41
3.4.3 Training algorithm selection (step 8).....	43
3.4.3.1 Levenberg – Marquardt.....	46
3.4.3.2 Resilient Backpropagation	48
3.4.3.3 Scaled Conjugate Gradient	50
3.5 Performance Reached (step 9)	52
3.6 TREPAN Module.....	53
3.6.1 Attributes definition (step 10).....	56
3.6.2 Options Setup (step 11).....	58
3.6.3 Data Definition (step 12)	59
3.6.4 Oracle Setup (step 13).....	59
3.7 Knowledge assessment (step 14)	62
3.8 Use of knowledge (step 15)	62
CHAPTER 4. EVALUATION OF THE INVENTORY KNOWLEDGE DISCOVERY SYSTEM	64
4.1 Time Series Forecasting of the demand.....	64
4.2 Impact of demand, lead time and cycle service level on safety inventory	69
4.3 Discussion and conclusions	75
CHAPTER 5. CASE STUDY	77
5.1 Problem Definition.....	80
5.2 Ordering Methods	81

5.3 Safety Stock at PRD.....	83
5.4 Inventory Level Forecasting at PRD.....	84
5.5 Discussions	90
CHAPTER 6. CONCLUSIONS AND FUTURE WORK	91
6.1 Summary of the findings.....	91
6.2 Contribution of the research.....	91
6.3 Limitations and Future Works	92
Author's Publications	93
References.....	94
APPENDIX.....	100
Appendix 1	100
Appendix 2.....	102
Appendix 3.....	105
Appendix 4.....	106
Appendix 5.....	109
Appendix 6.....	112

List of Figures

Figure 1: Research Areas	6
Figure 2: Supply Chain unit and inventory as a buffer	7
Figure 3: Efficiency boundary of Inventory level and Service Level.....	7
Figure 4: Influential factors of safety stock	9
Figure 5 Forecasting components	11
Figure 6: The AHP hierarchy for the demand forecasting process.....	15
Figure 7: Three parts of modeling real world	16
Figure 8: Knowledge Discovery Process	18
Figure 9: Data mining tasks	19
Figure 10: Comparison between artificial neuron and biological neuron.....	23
Figure 11: Simple Feedforward NN	26
Figure 12: NARX Network with T as time delay unit.....	29
Figure 13: Inventory Knowledge Discovery System (IKDS).....	36
Figure 14: Data acquisition	37
Figure 15: Data selection	38
Figure 16: Data preprocessing	38
Figure 17: Data transformation	39
Figure 18: FFNN (a) and NARX (b).....	39
Figure 19: Choosing number of input/output neurons.....	41
Figure 20: Symmetric hard limit transfer function (a) and linear transfer function (b).....	43
Figure 21: Log-sigmoid transfer function (a) and hyperbolic tangent sigmoid transfer function (b).....	43
Figure 22: Simple NN illustrating forward propagation of Inputs and backward propagation of Error.	44
Figure 23: Simplified flow of backpropagation algorithm	44
Figure 24: The pseudo-code of LM algorithm.....	47
Figure 25: Pseudo-code of RP algorithm.....	49
Figure 26: Pseudo-code for SCG algorithm.....	51
Figure 27: Pseudo-code of TREPAN algorithm	54
Figure 28: Example of attributes definition in TREPAN	57
Figure 29: Example of data definition table	59
Figure 30: Example of interpreted decision tree extracted by TREPAN.....	60
Figure 31: Use of knowledge in Inventory Management	63
Figure 32 Distribution of demand over time.....	64
Figure 33: Performance assessment for four NARX topologies (best validation performance obtained for 20 hidden neurons topology).....	66
Figure 34: Errors comparisons.....	67
Figure 36: Slopes representation for each value of demand	68
Figure 35: Proposed coding of slopes for time series case	67
Figure 37: Tree extracted by TREPAN for time series case.....	69
Figure 38: Relation between Service Level and Safety Factor	70
Figure 39: Continuous review.....	71

Figure 40: Training performance achieved by tested network topologies (best performance achieved by 20 hidden neurons topology)	73
Figure 41: Regression analysis for 20 neurons topology selected.....	74
Figure 42: Tree extracted by TREPAN	75
Figure 43: Information Flow in PRD.....	77
Figure 44: PRD purchasing process.....	79
Figure 45: Accuracy of forecasting in time	80
Figure 46: Monthly ordering.....	81
Figure 47: Weekly Ordering Method.....	82
Figure 48: Daily Delivery Ordering Method	83
Figure 49: Inventory, Production and Ordering for two items	83
Figure 50 :Inv. Level for item Hot Rolled Coil SQ -0231049.....	85
Figure 51: Interpolation function.....	85
Figure 52: Inventory level records chart.....	86
Figure 53: NN Performance analyses	87
Figure 54: Graphical representation of Error for P1	88
Figure 55: Graphical representation of Error for P2.....	89
Figure 56: Tree extracted for Case Study	89

List of Tables

Table 1: Intelligent technologies.....	31
Table 2: Transfer functions used.....	42
Table 3: Levenberg – Marquardt training parameters.	48
Table 4: Resilient Backpropagation training parameters.....	50
Table 5: Scaled Conjugate Gradient training parameters	52
Table 6: Performance functions	52
Table 7: Attributes definition.....	56
Table 8: Options used to control TREPAN algorithm.....	58
Table 9: TREPAN Tree interpretation.....	61
Table 10: Demand over 36 months.....	64
Table 11: RP training parameters for causal forecasting case	65
Table 12: Forecasted values.....	66
Table 13: Variable selection for time series case.....	68
Table 14: Sample from set used to train the NN.....	71
Table 15: Variable selection for TREPAN training.....	72
Table 16: Lead Times from suppliers	78
Table 17: Sample from PRD database	84
Table 18 :Monthly Inventory levels.....	85
Table 19: Training sample for P1	87
Table 20: Training sample for P2	88
Table 21 :Forecasting for P1 (2632)	88
Table 22: Forecasting for P2.....	88

CHAPTER 1. INTRODUCTION

1.1 Background

Over the past fifteen years, Inventory Management (IM) has become an important focus of competitive advantage for firms and organizations. The impact of IM has increased steadily, drawing on developments in management science, operation management, and information systems as knowledge discovery systems. To be successful in a continuously changing businesses environment nowadays, companies are required to invest in knowledge acquisition for maintaining of their operational efficiency; they have to renew and update knowledge continuously. Basic activities of knowledge management comprise collecting, processing, and use of knowledge. A necessary condition before the outcome of knowledge can be deployed is the availability of knowledge. Hereby discovery of knowledge plays a decisive role to increase the availability of knowledge, the capacity to act, and finally, the effectiveness of companies. This work contributes to the discovery and understanding of knowledge in Inventory Management field through an exploration of different approaches of training and extracting knowledge embedded in Artificial Neural Networks, trained on a prediction problem.

Process related to investigation and extraction of rules is Knowledge Discovery Process (KDP). Fast growth of Knowledge Discovery Process (KDP) is due to the contribution of multidisciplinary fields, initially from automation of scientific discovery [1] and machine learning [2] and later from statistics and database domains. Large advances to the development of this area have been published in two books about KDP, by Piatetsky-Shapiro [3] and Fayyad [4]. They have started the research in KDP and base the consolidation of the KDP research paradigm. Among all the tools used by KDP the current work focuses on Artificial Neural Networks (ANN or NN). An NN involves a network of simple processing elements called neurons which can approximate even the most complex dynamical functions by adjusting its weights, process regarded as learning. NN were first proposed by neurophysiologist Warren McCulloch, and Walter Pitts, a logician, in the year 1943. In our days NN are used with success in many fields like medicine, logistics, and manufacturing with the purpose to approximate functions, recognize patterns, classify data, cluster and predict future values.

1.2 Problem statement

Inventory management efficiency is affected considerably by the forecasting accuracy which can be enhanced if knowledge about its variables is available. A form of knowledge which is easy to understand by humans is decision trees. Whether the system is efficient depends on the quality of rules described in the form of decision tree. However, knowledge detection and acquisition is not so easy. In the inventory management field, some knowledge cannot be described easily by knowledge engineers, some cannot be understandable easily by managers, and some cannot be identified easily by intelligent systems. This has been already the bottleneck of knowledge engineering. Considered as a tool of machine learning, an artificial neural network has the capability to self learning hence obtaining rules by itself through the process of inductive machine learning, but rules obtained by neural networks are hidden in the architecture and weights of networks which are not easy to understand by human being, and not easy to be applied to improve inventory management [5].

1.3 Motivations

In almost all situations of industrial environments, decisions are made based on huge amounts of knowledge, which is often incomplete, and vague. The majority of managerial decisions can be described as inference processes, where they apply certain rules of inference to a given situation and to their acquired knowledge in order to achieve a certain goal. Human experts can perform a wide variety of inference extremely fast, but human resources in an organization are limited and expensive. On the other hand the amount of information in the world is estimated to double every 20 months. It is spread around data warehouses all around the world in different formats. For example in Singapore, Storage and Warehousing has increased from SGD 1005 M to 1266 M from 2004-2006, and value added from 752M to 876M, also the amount of information associated with it (*Transport and Storage Service, Economic Survey Series, 2006*).

Human experts are unable to analyze all the data that is accumulated in organizations databases, but computers can analyze it with help from Neural Networks (NN) tools. NN are information processing systems that imitate functionality of the human brain. Because

Chapter 1 Introduction

NN are powered by computers, they can analyze databases with the purpose of finding relations between, like identifying the independent variables on which other variables (dependent) can be predicted.

Some of the NN outstanding features that attract the academic and industry attention are described below [6-8]:

- Learning abilities (networks are modifying its weights in order to learn)
- Generalization aspects (NN can perform well in instances never seen before)
- Robustness and fault tolerance (NN are capable to work even if a malfunction of some neurons appears).

1.4 Objectives

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

The main objectives of this research are listed below:

- To examine industrial problems related with demand prediction, focusing on inventory management.
- To identify various types of traditional forecasting methods and NN techniques to forecast demand in inventory management.
- To identify how knowledge patterns can be detected using NN in databases.
- To investigate how knowledge can be extracted and formulate as knowledge rules.
- To proposed an improved or new knowledge discovery system in inventory management.
- To validate the findings using case study analysis.

1.5 Scope

This study covers fields as Logistics, Inventory Management, Forecasting, Knowledge Discovery Process, Data-mining, Neural Networks and Rule extraction. Inventory Management is the main industrial field from which the data is collected and analyzed in prediction issues. Two type of forecasting models are analyzed: causal forecasting and time series forecasting, therefore two NN architectures are selected which best fits the forecasting models: Feedforward networks (FFNN - for causal forecasting) and Nonlinear Autoregressive eXogenous networks (NARX - for time series forecasting)

The significance of this study could lead to improvements and understanding of the relationship between variables involved in forecasting process, and further, the relationship between industrial factors which are influencing the inventory management efficiency. A detailed survey of the fields covered by this study is made in literature review chapter.

1.6 Layout of the report

The remaining chapters of this thesis are organized as follows:

- 2 LITERATURE REVIEW** chapter provides the background information of which this work is based. The first section describes the influential factors of inventory management and safety stock. The second section describes the knowledge discovery process as the method that detects hidden relationships in data. Last two sections describe the fundamental concept of neural network and the rule extraction techniques. Artificial neural networks have the ability to learn facts about an environment by feeding the inputs with quantified observations data collected from industrial environment. Rule extraction techniques are used to extract knowledge embedded in NN weights.
- 3 DESIGN AND DEVELOPMENT OF A KNOWLEDGE DISCOVERY SYSTEM IN INVENTORY MANAGEMENT** chapter proposes a research methodology composed of 15 steps. The proposed framework uses techniques from Knowledge Discovery Process for data acquisition and transformation. Neural Network learns hidden relationships in data and the TREPAN algorithm extracts the knowledge embedded in trained NN.
- 4 EVALUATING THE KNOWLEDGE DISCOVERY SYSTEM** deals mainly with testing of the research framework proposed in Chapter 3. The proposed methodology is tested in two cases.
- 5 CASE STUDY** chapter analyzes the inventory policies and problems encountered at a company located in Singapore. An analysis of the inventory records is made and the proposed framework in Chapter 3 is used to evaluate and forecast the inventory level of an item of the company.
- 6 CONCLUSIONS and FUTURE WORK** chapter summarizes the findings and conclusions of the entire thesis, pointing the limitations of the proposed framework and proposes new directions to be studied.

CHAPTER 2. LITERATURE REVIEW

This chapter provides a literature survey of topics related with this work. For a better understanding and a clear delimitation of the fields on which this study focuses, Figure 1 was designed to depict the main research areas:

Figure 1: Research Areas

Literature review begins with an overview of inventory management and safety stock and the factors that are influencing its accuracy; one of the influential factors is forecasting accuracy of the demand, therefore the traditional methods used in literature to forecast are investigated. Next this study addresses knowledge discovery process as the engine of discovering knowledge hidden within data. The neural architectures used to forecast in inventory management and logistics are addressed with emphasis on Feedforward and NARX networks. In the last sections of the literature review, main methods for knowledge extraction from a train neural network are described. Chapter ends with the implications of the literature review.

2.1 Inventory management

Inventory Management is an essential component of Logistics, which is defined [9] as “*the process of planning, implementing, and controlling the efficient, effective flow and storage of goods, services, and related information from point of origin to point of consumption for the purpose of conforming to customer requirements*”. According to other authors [10] logistics is the nervous system of any Supply Chain and is defined as a set of value-adding activities while inventory serves as a buffer (Figure 2).

Figure 2: Supply Chain unit and inventory as a buffer

Superior performance can be achieved by taking an integrated view of logistics activities, divided in [10] into four main fields: (1) purchasing management (ordering policies, suppliers selection) , (2) manufacturing management (production capacity), (3) distribution management (warehousing capacity, transportation methods) and (4) inventory management (inventory policies, reordering points). Inventory Management (IM) is *the art of managing inventory* which is defined as a physical resource that a firm/company holds in stock with the intent of selling it or transforming it into a more valuable state [9]. Inventory is a major investment in most of the companies which highly influences the internal administration of companies, e.g. by allowing production levels to change easily or by protecting against uncertainty of demand and supply. Although inventory is an asset for companies, it also ties up working capital and space and it can suffer from obsolescence or deterioration. Determining the appropriate level of inventory is not an easy task. Therefore one of the primarily goal of IM is to reduce inventory level while keeping a high level of product availability (Figure 3).

Figure 3: Efficiency boundary of Inventory level and Service Level

To achieve high performance of IM, factors which influence the performance have to be identified. In work of [11], five IM performance influential factors are identified as following:

1. Technical factors- characterized by the absence of availability, reliability and knowledge of efficient technology
2. Organizational factors- characterized by the absence of right technical input and financial support
3. Financial factors- characterized by the absence of explicit financial mechanisms
4. Managerial factors - characterized by the absence of training, improper managing, and
5. Informational factors - characterized by the absence of appropriate information, and information sharing.

Another study on factors influencing IM performance conducted in work [12] has grouped IM performance factors under: (1) Economic, (2) Behavioral and (3) Organizational categories. Similarly in [13] IM influential performance factors are grouped into: (1) Safety stock planning of 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.

In all three works [11-13], inventory turnover is used to measure IM performance which reflects how frequently a firm flushes inventory from its system within a given financial reporting period. The reporting period can be any time interval as monthly, quarterly, or annually. The measure can be computed for any type of inventory, raw materials, work in progress or finished products and is determined using the following formula:

$$Inventory\ turnover = \frac{Cost\ of\ products\ sold}{Average\ inventory}; \quad (1)$$

Inventory turnover also has been used as an “accounting measure of performance” in studying the use of Just-in-Time techniques [14] or continuous monitoring an inventory control system [15]. A low inventory turnover may point to overstocking or obsolescence. In some instances a low rate may be appropriate, such as where higher inventory levels

occur in anticipation of rapidly rising prices or shortages. On the other hand, a high turnover rate may indicate inadequate inventory levels, which may lead to a loss in business.

Other researchers narrowed the number of IM performance factors to technical ones, and studied only the influential factors of Safety Stock (SS) which is the extra units of inventory carried as protection against possible stock-outs because of uncertainty of demand and lead time [16]. Some of the factors which are influencing SS performance are determined in [17], as lead time, number of customers, customer satisfaction, delivery reliability, supply reliability, [18] links to factors as use frequency, quality grade, and actual SS, while in [19] are brought forward factors as stockout cost, sales situation. Figure 4 is an integration of major influential factors that are determining SS.

Figure 4: Influential factors of safety stock

As a conclusion, researchers have identified many factors that are influencing IM performance such as the lead time, supply reliability, delivery reliability and forecasting accuracy. Forecasting accuracy is one of the most important influential factors of the safety stock, and therefore of the inventory management, as it is determined by external factors such as the number of customers, customer satisfaction, or internal factors as product quality and reliability.

Next section deals with demand forecasting methods used in literature with emphasis on time series techniques and causal techniques.

2.2 Demand Forecasting

Demand Forecasting is an essential process for managing inventory efficiently. It is an essential skill for all operations and manufacturing professionals, as well as those professionals in logistics. Forecasting is a complex issue, with many interactions among functions and forecast variables. A well-designed forecasting system can contribute significantly to Supply Chain performance, especially to Inventory Management which is dependent on the accuracy of the forecasting method. If the predicted demand is more accurate, then Safety Stock can be reduced to minimum, thereby reducing the inventory cost and total logistic cost.

In [20] are defined four basic requirements for inventory forecasting: (1) *projecting requirements during the order cycle*-because of the time interval between order placement and delivery, sufficient stock level must be maintained which means that requirements for both cycle and safety stock need to be determined by forecasting, (2) *forecasting by time period* – subjects to be covered in inventory and production planning are anticipation of future demands, planning of facility and equipment requirements, material and production capacity, (3) *indicating changes in demand*- forecasting should reveal and signal changes in demand patterns so that inventory requirements can adapt, and (4) *projecting for multiple items*-logistics forecasting should involve anticipating requirements for the entire product line.

According to [21], logistical forecasting covers the projection of customer demands by location, product, and time period. Logistical forecasting is based on analysis data, such as historical demand patterns, customer intelligence, and scheduled promotions and programs. Logistical forecasts consists of the following components (Figure 5): (1) the seasonal component which tracks upward or downward movement in the demand pattern, (2) trend component which is long range general movement in demand over a long period of time, (3) the cycle component which refers to wide swings in the demand pattern lasting usually a year or less, and (4) irregular component which covers random fluctuations in demand.

Figure 5 Forecasting components

The basic types of forecasts identified in [22] are long-term, medium-term and short-term forecasts. The long-term forecasts cover a time span of 3-10 years and they are used in the analysis of fixed commitments and requirements for new plant or warehouse capacity. Then medium-term forecasts are made for one year to support production planning in the face of highly cyclical demand or raw material supply. The short-time forecasts cover a time period of 1 week to 3 months and they are used to control manufacturing levels and stock replenishment in the face of short-term demand variations.

The number and variety of available forecasting methods and techniques makes the choice of the most appropriate method a complex process. In [23], the following criteria for the evaluation of the applicability of a certain method are identified: (1) accuracy, (2) the forecast time horizon, (3) the availability of data, (4) the type of data pattern, and (5) the experience of the practitioner at forecasting.

In [24] seven basic steps for selecting and implementing the appropriate forecasting technique are defined: (1) identify the problem or purpose to be addressed by the forecast, (2) gather available factual data covering both the internal and external environments of company, (3) determine which forecasting method is most compatible with the objectives of the company and the data type available, (4) generate good assumptions concerning each of the forecast elements with as high accuracy as possible, (5) compare the forecast to expectations which means reviewing the initial forecast and comparing its outcome with the results expected or with the actual result, (6) analyze variance, and (7) adjust the forecast in order to make it a more accurate reflection of reality.

According to [25] forecasting methods are classified in qualitative and quantitative:

1) Qualitative: Qualitative forecasting methods are subjective and rely heavily on human experts' judgment. Quantitative methods are suitable for situations when little historical data is available or is missing. Qualitative methods have a "strategic essence", in the sense that the analysts can forecast demand several years for a new industry, and they do not rely heavily on mathematical computations.

2) Quantitative: Quantitative methods are objective and based on quantitative models which rely heavily on mathematical computations. Quantitative methods includes: Simulation Models, Time Series Models and Causal Models. Simulation forecasting methods imitate the customer choices when purchasing a product. Using simulation, a firm/company can combine time-series and causal methods aiming a better forecasting model.

2.2.1 Time Series Models

Time-series methods 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 (1) systematic, seasonal variations in the data, (2) cyclical patterns, (3) trends, and (4) growth rates of trends. Usually traditional time series (TS) methods [26-31] assume that a signal is linear and can be depicted by a set of linear equations. Therefore methods include autoregressive (AR), AR with eXternal input series (ARX), Vector Auto-Regression (VAR [32]), Autoregressive Moving Average (ARMA), and Autoregressive Integrated Moving Average (ARIMA). ARIMA (p , d , q) is the most general class of models for forecasting a time series models which can be stationarized by transformations such as differencing and logging [33] and is characterized by: p represents the no of autoregressive terms, d represents the number of non-seasonal differences, q represents the number of lagged forecast errors in the prediction equation.

Some of the advantages of linear TS models are that: 1) they can be understood in great detail because they have been studied for a long time and 2) they are easy to implement. But because of the limitation of linearity, they cannot be applied to the complicated problems. On the other hand, nonlinear models, such as TAR (Threshold Autoregressive model) [34, 35] are used for nonlinear time series modeling. However, for

most TS, a priori models are unknown and it is difficult to obtain a good guess due to large number of variables used, the high level of noise, and the reduced amount of training data.

2.2.2 Causal forecasting

In some cases, the variable to be forecasted has a rather direct relationship with one or more 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 variables to develop a relationship between a lead event and the event being forecasted [21]. A typical example of causal methods is regression analysis. By using regression, the demand forecast is based on a correlation of one event to another. The reliability of regression-based forecasting can be increased by using cause-effect relationships. The use of regression analysis requires a large amount of data for the forecast variable.

In addition to forecasting methods, management should put emphasis on the process of forecasting [22]. The process of forecasting consists of the procedures used in developing and using forecast, such as the assignment of responsibility for forecast, the reconciliation of aggregate and individual item forecast, the approval of forecasts by different levels of management, the adjustments of forecasts by managers to reflect conditions not captured in a forecasting model, and the agreement on a single forecast by the different functional groups within an organization. One approach used in causal forecasting for indentifying and decomposing the influential factors of demand is to use an Analytic Hierarchy Process (AHP).

The Analytic Hierarchy Process is a theory of measurement for dealing with quantifiable and intangible criteria that has been applied to numerous areas, such as decision theory. AHP is a problem-solving framework and a systematic procedure for representing the elements of any problem [36]. AHP is based on the following three principles: decomposition, comparative judgments, and the synthesis of priorities. AHP starts by decomposing a complex, multi-criteria problem into a hierarchy where each level consists of a few manageable elements which are then composed into another set of elements. The second step is to use a measurement methodology to establish priorities among the elements within each level of hierarchy. The third step is to synthesize the priorities of the elements to establish the overall priorities of the elements.

In work of [37] three primarily areas in forecasting are suggested where AHP can be applied. Firstly, AHP can be used as an expert-option forecasting tool. The second application is to use AHP in the selection of the most appropriate forecasting method. Thirdly, AHP can be used to combine the results of several forecasting techniques to produce a single, composite forecast. Applications of AHP in forecasting can be found in [38]. In work of [39], a decision support system is proposed for demand forecasting (Figure 6). The process consists of three steps: (1) identify the factors affecting the demand level and create the structure of AHP hierarchy, (2) assign priorities to the elements in the hierarchy, and (3) synthesize the priorities to obtain the overall priorities for the elements and calculate the composite demand forecast. In step (1) the logistics executives define the following major factors and environmental forces to be considered in the forecasting process: the development of national economy of the market area, the major competitors, the present and potential customers, the development of ecological factors in the market area, and company itself. In order to reach a level of sufficient detail in the analysis, the actors and environmental forces are divided into sub-components. The elements of the last level of the hierarchy define the possible changes rates of demand for company's products compared with the sales estimate for the present year (the values are determined also by the logistics executives). In step (2) are derived priorities for the elements of the hierarchy. The priorities are set by comparing each set of elements in a pairwise fashion with respect to each of the elements in a higher stratum. A nine point scale or smaller is used for the comparisons and is based on subjective, qualitative data. After priorities are assigned for each factor of all levels of the hierarchy, in step (3) the forecasted demand is calculated.

One limitation of the AHP technique is that the priorities for the elements of the hierarchy are assigned in a qualitative manner, using human judgment. It would be desirable that the priorities to be assigned in an objective manner, by using quantitative computation instead of human reasoning. One possible solution is offered in this study by training a neural network to forecast demand, then use an algorithm to extract the relation learnt by the network and therefore interpret the relations and priorities between input variables. It can be stated that the neural network is used to find the hidden relationships between input variables used in the forecasting model. In the literature, the process used to

Chapter 2 Literature Review

find hidden patterns into data is the Knowledge Discovery Process described in next section.

2.3 Knowledge discovery Process

The process employed with discovering and acquiring of knowledge from unknown or potentially useful information is Knowledge Discovery Process, which in literature is considered as a task of modeling real-world phenomena [40]. Knowledge discovery is a process composed of three parts which is shown in Figure 7: Observations, model design and qualitative information processing.

Figure 7: Three parts of modeling real world

A human can observe and acquire qualitative information efficiently through cognitive process. On the other hand computers use quantitative methods to induce models from data. Usually in literature, the term inductive inference is used to describe model construction from observation (for example, predicting the next variable based upon a given series of variables, such as time series prediction). On the other hand, grounded theory [41] describes an extreme approach which assumes that there is no a priori knowledge available, but all the theories and models are constructed from observed data. This is characteristics to the artificial neural network (ANN) in which is very difficult to integrate *a-priori* knowledge.

The relation between a quantitative model and a user's knowledge is always a complicated problem and one should be consider the efficiency of methods in terms of their ability to give explanations. NN lacks on providing the explanations and further investigations are conducted in this study.

Knowledge Discovery Process was defined clearly in [42] as *a non-trivial process of identifying valid, novel, potentially useful and understandable knowledge in data.*

Researchers have proposed slightly different processes, a seven – step approach is described in Figure 8. A brief description of each step follows:

1) Goal identification. Goal identification has the purpose to clearly define what is to be accomplished. Goal identification is one of the most difficult tasks, as decisions about resource allocations as well as measures of success need to be determined. A clear problem statement is enlisted as set of criteria to measure success and failure.

2) Data selection - Data selection restricts subsets of data from larger databases and different kinds of data sources. The phase involves sampling techniques, and database queries.

3) Data Preprocessing – data preprocessing represents data coding, enrichment and clearing which involves accounting for noise and dealing with missing information. Usually the majority of data preprocessing takes place before data is permanently stored in a structure such as data warehouse.

4) Data transformation- has the purpose to change data into a form, from which useful knowledge can be extracted. Data transformation methods also reduce the dimensionality of data and eliminate statistical properties that have adverse effect on data mining techniques. Data transformation can be subdivided in *Data-normalization*, *Data Type Conversion*, *Attribute and Instance Selection*.

Data- Normalization involves changing numerical values of data so they can be reduced to a specific range. Neural Networks works well with numerical data scaled to a range between 0 and 1. Data normalization methods include enumerated Decimal Scaling, Min-Max Normalization, and Logarithmic Normalization. *Decimal Scaling* divides each numerical value by the same power of 10. *Min-Max Normalization* is commonly used in NN environments as the network learns more efficiently if inputs are small in dimensions. *Logarithmic normalization* presumes that replacing a set of values with their logarithms has the effect of scaling of values without loss of information.

Data Type Conversion is required as some data mining tools including neural networks and some statistics methods cannot process categorical data and vice versa. Some data mining techniques are not able to process numeric data in its original form.

Figure 8: Knowledge Discovery Process

Attribute and Instance Selection is carried out but some data mining algorithms have trouble with a large number of instances. To overcome these problems, one must make decisions about which attributes and instances to use when building the knowledge discovery model.

5) Data mining is one of the most important steps as it is the “heart” of the knowledge discovery process. In current work, data mining is based on an artificial neural network trained for prediction. Data mining is described more explicitly in subchapter 2.3.1

6) Evaluation and conclusion making is a user-oriented step where the analyst interprets the model, verifies its stability and validity, and evaluates the interestingness of the result. The model can be evaluated quantitatively with respect to data and formalized assumptions.

7) Use of knowledge The final objective of Knowledge Discovery Process is to apply knowledge in practice. Knowledge is the value asset in corporation and knowledge

deployment helps to solve the problems and improve the operation efficiency so as to achieve higher productivity.

2.3.1 Data Mining task and methods

Data-mining (DM) is one of the most important steps of Knowledge Discovery Process and has the purpose to analyze the data to find the hidden patterns, and systematic relationships between variables. Even though DM can be used in many ways to enhance industry processes and sales, this work focuses on Prediction; predictive data mining is the most common type of data mining technique and has the most direct business applications [43]. The taxonomy of data-mining by purpose is shown in generally in Figure 9. Mainly, data-mining is classified as unsupervised methods and supervised methods. Unsupervised learning is used to observe latent variables while in supervised learning make use of causal structure of link inputs to outputs. From unsupervised family methods, clustering is one of the most important tasks, while from supervised family methods, classification and prediction is one of the most representative tasks.

Figure 9: Data mining tasks

2.3.1.1 Clustering

Clustering can be considered as the most important *unsupervised learning* problem and it deals with finding a *structure* in a collection of unlabeled data. A suitable definition of clustering could be “the process of organizing objects into groups whose members are similar in some way”. In other words, *cluster* is a collection of objects which are “similar” between them and are “dissimilar” to the objects belonging to other clusters. A very useful

tool to cluster data is Self Organizing Maps (SOM) which can be used to solve free learning problems is developed in work [44]. This model enables the definition of a topological structure on the competition layer, so that adjacent cluster are represented by adjacent neurons and thus a similarity measure on the data can be represented more appropriately.

2.3.1.2 Classification

Classification is the ability of a system or algorithm to split or to order complex phenomena into a comprehensive and small parts, units or classes that serve for better understanding or to design new situations. Classification systems are typically designed by a structural analysis provided by tools from linguistics, concept theory and terminology, with the purpose to facilitate the retrieval or the understanding of the basic concepts of objects, facts or entities.

2.3.1.3 Prediction

The purpose of the prediction models is to determine future outcome rather than current behavior. In an academic context, prediction is defined as a rigorous statement forecasting what will happen under specific conditions, typically expressed in rules based forms IF A is true, then C will also be true, where A is the antecedent and C is the consequent. Mathematical models are frequently used to both describe the behavior of a system, and predict its future behaviors like regression.

Prediction can be considered as the most important task of data mining since it has an important impact on industrial fields like inventory management. For example demand for a certain item can be considered as a nonlinear function, and associated with stochastic functions. If a data mining prediction method is able to model demand function with acceptable errors, the overall inventory management efficiency is enhanced.

Basically, forecasting and prediction are used interchangeable and has similar meaning. Some researchers suggest that prediction provides information about the

probability of the predicted event, while forecasting offers less information. In this thesis, prediction and forecasting concepts are interchangeable.

After data-mining finds the hidden patterns of data used for prediction, it outputs the found relations in the form of rules or decision trees, described in next section.

2.4 Knowledge Representation

This subchapter covers types of knowledge produced by the data-mining techniques. The focus is on rules in the form of Decision trees as this form of representation is used later in methodology chapter.

2.4.1 Set of rules

Rules have the form “If A then C”, symbolically depicted $A \rightarrow C$, where A is called antecedent and C is called consequent. Rules are commonly used for knowledge representation, as are simple to represent and easy to be understood by humans. Two approaches are described briefly: Rough Set approach and Association Rules approach.

The theory of rough sets was introduced in work [45]. The idea behind this theory is derived from the simple fact in real life. When dealing with sets, we often have no means of distinguishing individual set elements. The elements may possess some measurable characteristics but in principle, due to limited solution of our perception mechanism, we can distinguish only classes of elements rather than individuals.

For example, an engine manufacturing company has an inventory database with lots of record. It is known the properties of the items are weight (W) which classified as low, medium and high, volume (V) as low, medium or high and horse power (HP) as low, medium, high. Regardless of how many records are in inventory database, it is assumed to have a maximum of eighteen classes of engines types corresponding to different combinations of W, V and HP. Each combination potentially corresponds to hundreds of engine models. It can be realized that engines have been classified into a certain number of categories, or elementary sets in rough set terminology. Such classification has several

advantages: 1) the ability to discover repetitive patterns in data since each combination potentially corresponds to a large number of engines and, 2) reduction of the complexity of information representation.

Association rule approaches are simple classes of sentences that can be efficiently discovered from large sets of binary data. These rules are not particularly powerful, but their usefulness stems from the ability of the algorithm is to find all associations rules satisfying certain conditions based on the existing data.

2.4.2 Decision-Trees

A decision tree is a rooted, directed acyclic graph consisting of a set of internal nodes and a set of leaves. Each internal node in a decision tree has an associated logical test based on the features in the domain. When classifying an example, the role of an internal node is to send the example down on one of the outgoing branches of the node. The decision as which to which branch an example is sent down is determined by the logical expression at the node. Decision trees are constructed using only those attributes best able to differentiate the concepts to be learned. Most common methods used in the literature are: C4.5, C5 (and advanced version of C4.5) and CART. First methods C4.5 and C5 belong to a succession of decision –a tree learner that goes back to the work of [46]. Input to C4.5 consists of a collection of training cases, each having a tuple of values for each set of independent variables (or attributes) $I = \{I_1, I_2, \dots, I_k\}$ and a class of dependent variables. An attribute I_a is described as continuous or discrete depending on whether its values are nominal or numerical. Class B is discrete and has values B1, B2, B3...Bx. The final goal is to learn from the training cases a function that maps from the attributes variables values to a predicted class. The other method is CART which stands from Classification and Regression Trees [47] and constructs trees that have only binary splits. For pruning, CART uses a technique called minimal cost complexity pruning, which assumes that the bias in the substitution error of a tree increases linearly with the number of leaf nodes.

2.5 Forecasting with Neural Networks

An Artificial Neural Network (ANN-Figure 10) is a parallel information processing system that has certain functionalities and performance characteristics in common with biological neuron networks. Artificial neural networks have been developed as generalizations of mathematical models of human neural biology.

Figure 10: Comparison between artificial neuron and biological neuron

A neural network is composed of one or more neurons, which is the basic processing element. A neuron has one or more inputs (dendrites), each of which is individually weighted. A neuron has one or more outputs (axon) that are weighted when connecting to other neurons. The neuron itself includes a function that incorporates its inputs (via summation) and then normalizes its output via a transfer function. These adjacent inputs are then summed and a transfer function is applied to determine the output. Basically all the computation of the network is made at the hidden layer and output layer by adjusting the weights and biases during the training process. Input layer only has the purpose to forward the signals to hidden layers.

The ability of the NN to learn nonlinear functions is determined by the nonlinearity of the activation functions. The equations that described the functionality of a neural network are depicted below:

$$v_j = \sum_{i=1}^d w_{ij} x_i + b_i \quad (2)$$

$$o_i = f(v_j) = f\left(\sum_{j=1}^d w_{ij} x_i + b_i\right) \quad (3)$$

$$w_{ij} = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{1m} \\ w_{21} & w_{22} & w_{23} & w_{2m} \\ w_{31} & w_{32} & w_{33} & w_{3m} \\ w_{k1} & w_{k2} & w_{k3} & w_{km} \end{bmatrix} \quad (4)$$

Where v_j is the linear combination of weight and inputs to hidden neuron j , b_i is the bias of neuron j and o_i is the output value of the hidden neuron j . w_{ij} is the weight matrix from input to the first hidden layer.

Next subchapter summarizes logistics fields in which neural networks have been applied with success.

2.5.1 Neural Networks in Logistics

Some of the Logistics fields in which Neural Networks have been successfully used as primarily solving methodology are summarized below:

- 1) **Optimization.** Optimization can be seen as the issue of constraints fulfillment. The optimization techniques can be applied in a large amount of activities which can be categorized as an optimization problem on Just-In-Time (JIT) transportation management, resource allocation and scheduling in both local and global perspective.
- 2) **Modeling and Simulation.** These are related with analyzing the dynamics of supply chain using techniques such as discrete event simulation and dynamic systems theory. NN has been widely used in control and modeling applications where classical system theories have failed to deliver satisfactory solution. The use of neural networks in the modeling and simulations of supply chain dynamics could lead to new understandings, hence allows more accurate predictions on the system behavior.
- 3) **Globalization.** Continuously increasing coordination of activities between firms has created an increased demand on information automation to globalized level. The use of

information technology such as Electronic Data Interchange (EDI) and Value Added Network (VAN) to shorten the time required to transfer information between centers has proved important to improve supply chain performance. Distributed Artificial Intelligence (DAI) is an increasing technology used to realize globalization of the companies. NN can be easily adapted to the DAI architecture if it is simply considered as a blackboard by producing output in response to what has been presented in the input.

4) **Decision Support.** Considering a SC where data is spread across a network in a distributed format, rigid data indexing scheme for data retrieval is clearly insufficient. The unique abilities of NN in data classification and self organization make it as an ideal tool to complement the conventional database query techniques.

5) **Forecasting.** Because of numerous successes on NN in predicting the performance of the financial markets, forecasting is perhaps the most discussed application area on NN from the users' point of view. The use of NN in forecasting can be described intuitively as follows. Considering that exists certain amount of historical data which can be used to analyze the future behavior of a system, then such data can be used to train a NN to correlate the system response with respect to time and other system parameters. From the industry point of view, the ability of NN to be integrated with the existing technologies is extremely important because it allows for incremental implementation which is a common strategy for introducing any new technology.

Some of the important issues addressed in the literature and related to neural networks in time series modeling include incorporating temporal information, selecting input variables and balancing model bias/variance trade-off.

Neural Network with hidden layers is universal approximators, which means that, in theory, they are capable of learning an arbitrarily accurate approximation to any unknown function, provided that they increase in complexity at a rate approximately proportional to the size of the training data. Neural networks can be applied to time series modeling without assuming a priori function forms of models. A variety of neural network techniques have been proposed, investigated, and successfully applied in prediction described in the following sections.

2.5.2 Multilayer Feedforward Neural Networks

The use of feedforward networks (Figure 11) in forecasting can be described intuitively as follows. Given certain amount of historical data which can be used to analyze the behavior of a particular system, such data can be used to train a NN to correlate the system with respect to time or other system parameter. Even though this seems a simplistic mechanism, experience shows that NN approach is able to provide a more accurate prediction than expert systems or statistical counterpart [48].

Figure 11: Simple Feedforward NN

Neural Networks usually becomes involved in forecasting lumpy demand, characterized by intervals in which there is no demand and periods in which demand has a large variation in level. Traditional time-series methods may not capture the nonlinear pattern in data. Therefore, NN modeling is a logical choice to overcome these limitations.

The following research papers adopted NN in order to forecast lumpy demand for spare parts, with very promising results. In [49] the authors adopted the most widely used method, multi-layered Perceptron (MLP) trained by a back-propagation algorithm (BP). The input neurons represent two variables: a) the demand at the end of the immediately preceding period and b) the number of periods separating the last two nonzero demand

transactions as of the end of immediately preceding period. The output node represents the predicted value of the demand for the current period. They used a momentum factor of 0.9 and a learning rate of 0.1.

In another paper [50] authors used a Generalized Regression NN (GRNN). This network does not require an iterative training procedure as in back propagation method. The input variables for GRNN are:

- a) Demand at the end of the immediately preceding target period.
- b) Number of unit periods separating the last two nonzero demand transactions as of the end of the immediately preceding target period.
- c) Number of consecutive period with no demand transaction immediately preceding target period.
- d) The mean of demand for four period immediately preceding target periods.

Another research [51] proposed the use of Recurrent Neural Networks (RNN). A RNN is basically a FFNN with the outputs forwarded back to the inputs. The network is composed of four layers: an input layer, a hidden layer, a context layer and an output layer. Real data set of 30 types of spare parts from a petrochemical company is used in the study. The following variables have been defined for the input layer:

- a) Demand at the end of the immediately preceding target period.
- b) The number of consecutive period with no demand transaction, immediately preceding target period.
- c) The number of consecutive periods with demand transaction, immediately preceding target period.
- d) The number of periods separating the last two nonzero demand transactions as of the end of the immediately preceding target period.
- e) The number of periods between target period and first nonzero demand immediately preceding target period.
- f) The number of periods between target period and first zero demand immediately preceding target period.
- g) The mean of the demand for six periods immediately preceding target period.
- h) The maximum demand among six periods immediately preceding target period.

A tan-sigmoid and saturated linear transfer functions in the hidden and output layer is used. To find the best number of neurons in the hidden layer, a range between 1 to 15 neurons has been tested and the one with the minimum error is selected. Backpropagation algorithm had been used to train the network, with a learning rate of 0.01.

A very interesting approach is proposed in [52]. The authors combined two methods to forecast: ARIMA and Neural Networks, the result is a hybrid model which combines the features of both approaches. Input neurons are divided in two types of variables: Binary variables (12 variables, e.g. payment, holiday) and simple numerical representing past sales with lag k (k – depends on the time window used).

2.5.3 NARX Neural Network

The Nonlinear Autoregressive eXogenous Neural Network (NARX) is a recurrent neural architecture commonly used for input –output modeling of nonlinear systems. The input of NARX is formed by two tapped-delay lines shown in Figure 12, one sliding over the input signal and other over the output signal. NARX network is trained basically under one out of the two modes:

1. *Series –Parallel Mode*. In this case, the output's regressor is formed only by actual values of the system's output.
2. *Parallel Mode*. In this case, estimated outputs are fed back and included in the output's regressor.

In [53] a comparison between NARX NN, Time Delay NN, and Recurrent NN (Elman Network) is made. The findings of their research is summarized as:

1. Long Term Dependence occurs very often in real-world time series (e.g. traffic series).
2. Theory of Dynamical Systems provides the theoretical bases to analyze nonlinear systems with chaotic behavior.
3. Recurrent Neural Networks are capable of representing arbitrary nonlinear dynamical mappings, such as those commonly found in nonlinear time series prediction.
4. NARX Model is a recurrent neural network capable of modeling efficiently time series with long-term dependences.

The results show that NARX network can be successfully applied to complex univariate time series modeling and prediction tasks. NARX approach consistently outperforms standard neural network based predictors, such as Elman architectures.

Figure 12: NARX Network with T as time delay unit

In work [54] and [55], a NARX network is used to deal with the problem of learning long-term dependencies, i.e., when the desired output depends on inputs presented at times far in the past, which has been shown to be a difficult problem to learn for gradient-based algorithms(gradient-descent learning algorithm is used in training RNN) . They found that although NARX networks do not circumvent this problem, it is easier to discover long-term dependencies with gradient descent in NARX architectures than in architectures without output delays. This has been observed previously, in the sense that gradient-descent learning appeared to be more effective in NARX networks than in recurrent neural-network architectures that have “hidden states” on problems including grammatical inference and nonlinear system identification.

In [56] long-term series prediction with NARX networks is investigated. She shows that NARX networks can successfully use their output feedback loop to improve their predictive performance in complex time series prediction tasks.

A few techniques to choose the topology of the network are described below.

2.5.4 Techniques used for variable selection.

Determining parameters for a NN is a very heuristic task; the past research papers are not clear in offering clear methods in choosing the suitable number of hidden layers and number of neurons contained in it. Some of the heuristics techniques are depicted below:

2.5.4.1 Sensitivity Analysis (I/O selection)

The selection of NN input variable is described as one of the most important steps of modeling process because the number of input variables used during the learning affects the quality of training data needed for learning. Both data collection and learning process can be expensive and time consuming. Two types of feature selection approaches are described below [57]:

- The model-independent approach, where statistical merits of input variables, for example, joint mutual information of input variables and target variables, are directly evaluated using data
- The model-dependent approach, which evaluates input features together with learning algorithms. A learning algorithm is applied to the data, and the subset of input features that gives the best result for the algorithm is used for modeling.

Sensitivity-based input variable selection techniques are model-dependent algorithms that prune unnecessary or noisy input variable from a training network.

2.5.4.2 Committees of Predictors

A method to deal with instability and noise during neural network modeling is to train a set of neural networks that have different initial weights, training parameters, or architectures, and then choose the network that performs the best on a validation data set. An alternative approach is to use committees. A committee consists of a set of member models. Committee approach has been used to reduce model variance and improve generalization performance. For committees to achieve effective generalization performance improvement over single models, committee members need to satisfy two

Chapter 2 Literature Review

criteria [58] - have reasonable individual performance and make decisions as independently as possible.

Next subchapter deals with methods used in literature to extract knowledge learnt by a trained neural network.

2.5.5 Rule extraction from trained neural networks

From previous chapter, it can be concluded that NN is able to learn interesting relations between input variables. They support learning but lack in explanation abilities. Table 1 denotes a comparison between different intelligent technologies used in literature [59].

Table 1: Intelligent technologies

	ES	FS	NN	GA
Knowledge representation	○	●	□	■
Uncertainty tolerance	○	●	●	●
Imprecision tolerance	□	●	●	●
Adaptability	□	■	●	●
Learning ability	□	□	●	●
Explanation ability	●	●	□	■
Knowledge Discovery and Data-mining	□	■	●	○
Maintainability	□	○	●	○
Rather BAD ■	BAD □	Rather GOOD ○	GOOD ●	

ES	Expert Systems
FS	Fuzzy Systems
NN	Neural Networks
GA	Genetic Algorithms

It becomes apparent that knowledge extraction and representation are essential for a better explanation of the models learnt by a network.

The induction of rules using an already trained NN is one of the important issues of data mining. Currently in the literature, 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* [60].

- First approach **decompositional**, 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 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, regardless of the activation on the other incoming links, have to be determined to construct local rules for the respective unit. Next, a rule base for the NN can be constructed by aggregation of the local rules. Examples of these methods are the KL algorithm developed in [61] and the SUBSET algorithm developed in [62]. Subset algorithms search for subsets of incoming weights that exceeds the bias on a unit. The search is made first at the single links (smallest subsets) that guarantee that the bias is exceeded. Next the size of the subsets is increased until all possible subsets have been analyzed and finally subsumed, and overlie general rules are extracted. The decompositional approaches are usually combined with a special network structure and learning technique, which ensures the quality of the extracted rules [63].

- On the other hand, **pedagogical** approaches treat the network as a black box. The rule extraction process is constructed as a learning task, which uses the training network just to generate examples. This technique is typically used in conjunction with symbolic learning algorithm. Some examples of such methods are the VI- analysis (VIA) techniques developed in [64], which uses validity intervals to derive the rules, with RULENEG algorithm [65].

It can be considered also a third approach, **eclectic** approaches, which combines decompositional and pedagogical methods [66] and focuses on extracting rules that include the most important input units.

As an alternative to IF-THEN-ELSE rule is M-of-N form. Rules in this form state that “if M of N conditions $x_1, x_2, \dots, x_m$ are true, then the conclusion y is true”. It is argued [67] that some concepts can be better expressed in such form, and use of this representation

also helps avoid the combinational explosion in tree size found with if-then rules. The following steps are performed in order to extract M-of-N rules:

1. With each output node, form groups of similar weighted links.
2. Set the link weights of all group members to the average of that group.
3. Prune any group that does not significantly affect the output value.
4. Optimize the biases of all output nodes, using the back-propagation algorithm, while holding weights constant.
5. Form a rule for each output node, using the weights and threshold of the rule.
6. If possible, create an M-of-N rule.

A flexible data-mining algorithm operating on NN and using the M-of-N rule representation would be therefore useful. The TREPAN algorithm [68], which does not require a specialized NN architecture or training, meets these requirements. This algorithm is so flexible that it is not restricted to feedforward NN 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 input space.

One output of TREPAN is a decision tree. The topology of the tree is represented in terms of nodes, parents, children, leaves and forks. Binary splitting occurs at each of the nodes, so that each node has either zero or two children. A node with no children is a leaf while a node with two children is a fork. A second output is a set of M-of-N rules, where the integer N is the total number of conditions and the integer M is the number of those that need to be satisfied for the rule to apply.

2.6 Implications from the Literature Review

In summary, the review of contemporary publications indicates that whilst many research studies have been conducted on using various approaches to discover knowledge, the research related to knowledge discovery for inventory management has not received the *attention* it deserves. In literature review, it has been identified influential factors of IM and safety stock, but there is not enough study in depicting which factors are more important in comparison to others.

To solve these issues, NN is regarded as the promising technique for knowledge discovery for classification, clustering and prediction, but it lacks the explanation abilities. It would be advantageous to explain the knowledge from the trained neural networks with decision trees and rules, which are comprehensible forms of knowledge representation, The reasons of adopting the proposed approach are:

- To provide explanations of models and patterns learnt by the NN.
- To improve the generalization ability of neural networks by using an extracted set of rules to predict where the data space the NN may perform badly. It is suggested that the regions of data space where new data should be collected and used to retrain the network.
- To automate and overcome the knowledge acquisition bottleneck for symbolic AI systems.

Probably the most important reason for extracting decision trees and rules from NN is to facilitate knowledge acquisition. As observed in [68], “a learning system may discover salient features in the input data whose importance was not previously recognized”. In this case, a fundamental question is, why do we construct decision trees from trained NN, when symbolic data mining techniques such as C5 and CART exist? Clearly, data mining with NN is worthwhile only if some increase in performance is observed, such as:

- The decision tree or rule set extracted by the NN technique gives a better fit to test data than symbolic techniques.
- The decision tree or rule set extracted by the NN techniques is smaller than produced by symbolic techniques, with the NN –based technique giving superior fit to test data than symbolic techniques.
- The decision trees or rule set extracted by NN technique is represented in more comprehensible way.

The next chapter proposes a research methodology which integrates steps from fields as Knowledge Discovery to deal with data acquisition and preparation. Neural Networks is used to map inputs with outputs variables in learning process (in a prediction task) and Rule extraction algorithms is adopted to understand the models learnt by network.

CHAPTER 3. DESIGN AND DEVELOPMENT OF A KNOWLEDGE DISCOVERY SYSTEM IN INVENTORY MANAGEMENT

In this chapter, a Knowledge Discovery System is proposed to acquire knowledge related to inventory management. There are three modules in the proposed system and the algorithms and methods which are composing the IKDS are analyzed in detailed in Section 3.1 to 3.8.

3.1 Inventory Knowledge Discovery System (IKDS)

This section begins by proposing a framework of Inventory Knowledge Discovery System (IKDS) (Figure 13) composed mainly of three modules: Data Preparation module, NN module (NN) and TREes PARrotting Networks (TREPAN) module. The framework is composed of fifteen steps, counted 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 selecting and preprocessing of raw data contained into a database, Neural Network module is used to learn hidden relations of data used to forecast, and TREPAN module is used to extract the knowledge acquired and embedded in NN.

The neural network is trained using the Backpropagation algorithm. There are many types of backpropagation but in the current proposed framework three versions of the backpropagation algorithm are selected to train the network: Levenberg-Marquardt (LM), Resilient backpropagation (RP) and Scaled Conjugate Gradient (SCG). The suitable algorithm is selected with respect of the learning task. LM converges faster and with high performance for function approximation problems (or linear regression). RP is better in pattern recognition problems (or nonlinear discriminant analysis) and SCG algorithm is used for both function approximation and pattern recognition problems but for networks with large numbers of inputs and hidden layers (hidden neurons), therefore with large number of weights (starting from 300 -400 weights).

TREPAN algorithm is composed of four steps: *Attributes Definition* which represents basically the variables used to train the NN; *Options Setup*, a set of values to control the efficiency and the output of the algorithm; *Data Definition*, which represents a

Chapter 3 Design and Development of a Knowledge Discovery System in Inventory Management

preprocessing of the data used by the TREPAN algorithm to extract the decision trees and lastly, *Oracle Setup* represents a build function which models the trained NN.

Figure 13: Inventory Knowledge Discovery System (IKDS)

3.2 Goal definition (step 1)

Defining the goal is one of the most difficult parts in knowledge discovery systems. In this thesis, two main goals are set. To train a NN to approximate a function (nonlinear regression) or to recognize a pattern (nonlinear discriminant analysis) for a prediction task, and the second goal is to find previously unknown relations between input variables, therefore to better understand the available data. Knowledge extracted in the form of decision trees, by TREPAN module, from trained NN is used to enhance inventory management.

3.3 Data Preparation module

3.3.1 Data acquisition (step 2)

This step deals with searching, identifying and purchasing real data contained in inventory databases from different industrial players and companies (Figure 14). Data can be acquired in the form of electronic tables or/and questionnaires and has different formats like excel (see Case study chapter), mysql, sql-server, oracle or simple text tables. If the tables or questionnaires are available in hard copies, transformation in an electronic form is mandatory.

Figure 14: Data acquisition

3.3.2 Data selection (step 3)

Not all the data sample acquired is relevant for analysis or is related to inventory management. The phase involves database queries and sampling techniques aiming the extraction of relevant attributes (variables) from database. Data selection step eliminates unwanted or parasite data from the sample analyzed. For example in Figure 15 database contains four attributes (Atr1, Atr2, Atr3 and AtrN) each having a values val1, val2, val3, and valN. After data selection only Atr1 and Atr3 are kept for the next step.

Figure 15: Data selection

3.3.3 Data preprocessing (step 4)

This step 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, this way the performance of the network is enhanced. For example in Figure 16 the value of Atr3 is missing. After preprocessing step, Atr3 is filled in with val3. One method to fill in the missing values of an attribute is by averaging the previous and the next neighbor values, if they exist. For example if Atr3 has m values and the value k th is missing then $val_k = \frac{val_{k-1} + val_{k+1}}{2}$;

Figure 16: Data preprocessing

3.3.4 Data transformation (step 5)

This step changes data into a form suitable for the inputs into NN. Not all the data available in IM are suitable to feed the inputs of a NN. For example, categorical data like TRUE or FALSE cannot be used as inputs or outputs. Therefore, numerical values like 1 and 0 are used to code TRUE and FALSE values. Data transformation step also can reduce the dimensionality of data by mapping row minimum and maximum values to $[-1 \ 1]$ as a NN learns more efficiently if input variables are small in dimension. In Figure 17 Atr1 has a categorical value F (False) while Atr3 has T (True). Transforming data, F becomes 0 while T becomes 1. Now data is ready for the next step, which is NN architecture selection.

Figure 17: Data transformation

3.4 Neural Network Module

3.4.1 NN Architecture selection (step 6)

In this research two Neural Network architectures are used: Feedforward Neural Network (FFNN-Figure 18a) and Nonlinear Autoregressive Network with Exogenous inputs (NARX Figure 18b). Choosing which 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 a FFNN is preferred. Both architectures can be used interchangeable but with less performance. The equation for the FFNN model is defined as following:

$$y(t) = f(u(t)), \text{ where } u(t) \text{ is the input and } y(t) \text{ is the output of the network}$$

The defining equation for the NARX model is:

$y(t) = f(y(t-1), y(t-2), \dots, y(t-n_y), u(t-1), u(t-2), \dots, u(t-n_u))$, where the next value of the dependent output signal $y(t)$ is regressed on previous values of the output signal and previous values of an independent (exogenous) input signal. In fact, a NARX network is a FFNN plus one or more Time Delay Line (TDL) at the input and recurrent line.

Figure 18: FFNN (a) and NARX (b)

3.4.2 NN Topology selection (step 7)

Topology selection deals with selecting the number of inputs of the network, outputs, number of hidden neurons in the hidden layer, number of hidden layers and types of the activation function.

3.4.2.1 Number of input/output neurons

The number of input in NN is equal with the number of variables. In the case of time series models only one input is available and only one output neurons. In time delay and NARX networks, the main issue is choosing the time delay units. If the number of time delay is small, the memory of the network is small.

3.4.2.2 Number of hidden layers and neurons in the hidden layer

Unfortunately, there is no clear theory in choosing the number of hidden neurons or the number of hidden layer. This is done heuristically by varying the number of hidden neurons until the network is able to learn satisfactory. Larger numbers of neurons in the hidden layer give the network more flexibility because the network has more parameters it can optimize. Increasing of the hidden layer is made gradually. If the hidden layer is too large it can cause the problem to be under-characterized and the network must optimize more parameters that there are data vectors to constrain these parameters. It has been demonstrated that usually only a single hidden layer is sufficient to approximate even the most complicated functions. Therefore only one hidden layers is selected in this thesis. If the network is unable to be trained, the number of hidden neurons is increased gradually. If the network is still unable to be trained, the number of hidden layers will be increase (Figure 19).

Figure 19: Choosing number of input/output neurons

One heuristic that deserves to be mentioned relates to the size of the training set, Tr_size , for a pattern recognition task. Given a multilayer FFNN with a total number of synaptic weights including bias levels, denoted by W , a rule of thumb for selecting Tr_size is:

$$Tr_size = Or_of\left(\frac{W}{E}\right) \quad (5)$$

Where Or_of denotes “the order of”, and E denotes the fraction of recognition errors permitted on the test data. For example, with an error of 5%, the number of training examples needed should be five times the number of synaptic weights in the network.

The ability of network to approximate nonlinear function is powered by the type of the activation function. Several activation functions are used in this thesis.

3.4.2.3 Activation function selection (Transfer function)

Basically, four types of activation functions are commonly used in training a network, for function approximation or pattern recognition problem which is depicted in Table 2:

Table 2: Transfer functions used

Symmetric hard limit transfer function Algorithm: $f(wx + b) = \begin{cases} 1, & \text{if } (wx + b) \geq 0 \\ -1, & \text{otherwise} \end{cases} \quad (6) \quad \text{or}$ $f(wx + b) = \begin{cases} 1, & \text{if } (wx) \geq -b \\ -1, & \text{otherwise} \end{cases} \quad (7)$	Linear transfer function Algorithm: $f(wx + b) = wx + b$
Log-sigmoid transfer function Algorithm: $f(wx + b) = \frac{1}{1+e^{-(wx+b)}} \quad (8)$	Hyperbolic tangent sigmoid transfer Algorithm: $f(wx + b) = \frac{2}{1+e^{-(wx+b)}} - 1 \quad (9)$
Where : $w*s$ is product matrix of weights matrix and inputs matrix, and b is the bias of corresponding neuron	

The power of the neural network to model nonlinear functions is assigned by the type of the activation function used. In this work ,the hidden layer is used a hyperbolic tangent sigmoid function and in the output layer a linear transfer function. Usually a network with biases, a sigmoid hidden layer, and a linear output layer is capable of approximating any function with a finite number of discontinuities. Sigmoid activation functions are preferred because their derivative is easy to calculate in a computational environment. The bias has the purpose to shift the activation value wx with b (consider the case of a symmetric hard limit transfer function). The value of b is changing during the training process, making the activation function more flexible in adjusting to the learning problem.

Figure 20: Symmetric hard limit transfer function (a) and linear transfer function (b)

Figure 21: Log-sigmoid transfer function (a) and hyperbolic tangent sigmoid transfer function (b)

3.4.3 Training algorithm selection (step 8).

All the training algorithms used in this research is based on Backpropagation (or Delta Rule). The Backpropagation algorithm is a typical supervised learning algorithm, where input vectors and the corresponding target vectors are used to train a neural network until it can approximate a function or recognize a pattern. Basic backpropagation is a gradient descent algorithm, during training the weights and biases are updated in the

direction in which the performance function decreases most rapidly (Figure 22). A simplified flow of backpropagation algorithm is described in Figure 23:

Figure 23: Simple NN illustrating forward propagation of Inputs and backward propagation of Error.

Figure 22: Simplified flow of backpropagation algorithm

In Figure 23, MSE stands for *Mean Square Error* and is the typical performance function used for training feedforward networks being defined as:

$$MSE = \frac{1}{N} \sum_{i=1}^N (\delta_i)^2 = \frac{1}{N} \sum_{i=1}^N (t_i - a_i)^2 \quad (10)$$

where δ_i is the error of Output node i ,

t_i represents TARGETS and a_i Actual_Output,

N is the number of examples used during training process.

Chapter 3 Design and Development of a Knowledge Discovery System in Inventory Management

One iteration of the algorithm can be written as:

$$w_{k+1} = w_k - \alpha_k g_k \quad (11)$$

where w_k is a vector of the current weights and biases,

α_k is the learning rate and g_k is the current gradient.

The rules governing the changing of weights [69] at each input/output pair p is given by:

$$\Delta_p w_{ji} = \alpha (t_{pj} - a_{pj}) i_{pi} = \alpha \delta_{pj} i_{pi} \quad (12)$$

where t_{pj} is the target input for j th component of the output pattern for pattern p ,

a_{pj} is the j th element of the actual output pattern produced by the presentation of input pattern p ,

i_{pi} is the value of the i th element of the input pattern,

and $\Delta_p w_{ji}$ is the change to be made to the weight from the i -th to the j -th neuron following presentation of pattern p . The weight on each line should be changed by an amount proportional to the product of an error signal, δ , available to the unit receiving input along that line and the output of the unit sending activation along that line:

$$\Delta_p w_{ji} = \alpha \delta_{pj} \quad (13)$$

The determination of the error signal is a recursive process which starts with the output units. If a neuron is an output, its error it is given by:

$$\delta_{pj} = (t_{pj} - a_{pj}) f'_j \left(\sum_i w_{ji} a_{pi} \right) \quad (14)$$

where $f'_j \left(\sum_i w_{ji} a_{pi} \right)$ is the derivative of the activation function which maps the total input to the unit to an output value. The error signal for hidden neurons for which there is not specific target is determined recursively in terms of the error signal of the neurons to which it directly connects and the weights to those connections. If the unit is a hidden neuron then:

$$\delta_{pj} = f'_j \left(\sum_i w_{ji} a_{pi} \right) \sum_k \delta_{pk} a_{kj} \quad (15)$$

There are two different ways in which the gradient descent algorithm can be implemented: incremental mode and batch mode. In incremental mode, the gradient is computed and the weights are updated after each input is applied to the network. In batch mode the weights and biases of the network are updated only after the entire training set has been applied to the network. The gradients calculated at each training example are

added together to determine the change in the weights and biases. In this work batch mode is chose for training the network as is more efficient in learning tasks with poor data available.

There are many variations of the backpropagation algorithm but in this thesis we are focusing only on three of them: Levenberg – Marquardt (LM), Resilient Backpropagation (RP) and Scaled Conjugate Gradient. Levenberg – Marquardt will be used to function approximation problems (or nonlinear regression), Resilient Backpropagation (RP) algorithm to pattern recognition problems (or nonlinear discriminant analysis). Scaled Conjugate Gradient is used in both cases when the number of weights is high (few hundred). SCG is useful because the performance of LM and RP is decreasing when the number of weights is high.

3.4.3.1 Levenberg – Marquardt

Levenberg – Marquardt [70] is the fastest algorithm for function approximation problems. The algorithm was designed to approach second-order training speed without having to compute the Hessian matrix (is the square matrix of second-order partial derivatives of errors with respect to weights). When the performance function has the form of a sum of squares, then the Hessian matrix (H) can be approximated as:

$$H = J^T J \quad (16)$$

And the gradient(g) is computed as:

$$g = J^T e \quad (17)$$

where J is the Jacobian matrix (is the matrix of all first-order partial derivatives of errors with respect to the weights) that contains first derivatives of the network errors with respect to the weights and biases,

and e is the vector of network errors. Therefore the update of the weights is made as:

$$W_{k+1} = W_k - [J^T J + \mu I]^{-1} J^T e \quad (18)$$

Where e is the error, J the Jacobian matrix and μ (mu) is the learning rate.

The standard LM training process can be illustrated in the following pseudo-code (Figure 24):

1. Initialize random weights and parameter μ .
2. Compute *sum* of the squared errors over all inputs $F(w)$. { $F(w)$ is the performance index $= e^T e$ and $w=[w_1, w_2, \dots, w_N]$ consists of all weights of the network }
3. Solve $\Delta w = [J^T J + \mu I]^{-1} J^T e$ { Δw is the increment of weights }
4. Recomputed the *sum* of squared errors $F(w)$ using $w + \Delta w$ as the trial w , and judge
IF trial $F(W) < F(W)$ in step 2 **THEN**
 $W = w + \Delta w$
 $\mu = \mu * \mu_{inc}$ { μ_{inc} }
GO to step 2.
ELSE
 $\mu = \frac{\mu}{\mu_{dec}}$ { μ_{dec} }
GO to step 4.
END IF

Where Jacobian matrix is defined as

$$\begin{bmatrix} \frac{\partial e_{11}}{\partial w_1} & \dots & \frac{\partial e_{11}}{\partial w_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial e_{kp}}{\partial w_1} & \dots & \frac{\partial e_{kp}}{\partial w_n} \end{bmatrix};$$

Figure 24: The pseudo-code of LM algorithm

The parameters used in training the neural network using LM algorithm are described in Table 3.

Table 3: Levenberg – Marquardt training parameters.

Parameter	Description
<i>epochs</i>	The training stops if number of iterations exceeds <i>epochs</i> parameter.
<i>goal</i>	Training stops if the performance function drops below <i>goal</i> parameter.
<i>min_grad</i>	Training stops if the magnitude of the gradient is less than <i>min_grad</i> parameter.
<i>max_fail</i>	Used for early stopping.
<i>mu</i> [μ]	Initial value of μ .
<i>mu_inc</i>	Multiply <i>mu</i> whenever a step increases the performance function.
<i>mu_dec</i>	Multiply <i>mu</i> whenever the performance function is reduced by a step
<i>mu_max</i>	Training stops if <i>mu</i> becomes larger than <i>mu_max</i> parameter.
<i>mem_red</i>	Used to control the mount of memory used by algorithm. (if <i>mem_red</i> is 2 then only half of the Jacobian is computed at one time)
<i>time</i>	Training stops if training time is longer than <i>time</i> parameter (seconds).

3.4.3.2 Resilient Backpropagation

Resilient backpropagation (RP) [71] is the fastest algorithm for pattern recognition problems. Sigmoid functions are characterized by the fact that their slopes must approach zero as the input gets large. This causes a problem when steepest descent is used to train a multilayer network with sigmoid functions. The gradient can have a very small magnitude and, therefore, cause small changes in the weights and biases, even though the weights and biases are far from their optimal values. The purpose of the resilient backpropagation (RP) training algorithm is to eliminate this adverse effects on the magnitudes of the partial derivatives. Only the sign of the derivative can determine the direction of the weight update; the magnitude of the derivative has no effect on the weight update. Whenever the weights are oscillating, the weight change is reduced. If the weight continues to change in

the same direction for several iterations, the magnitude of the weight change increases. The pseudo-code of RP algorithm is described in Figure 25:

```

 $\Delta_{wij} = -\text{sign}(\partial E / \partial w_{ij}) \Delta_{ij}$  {  $\Delta_{wij}$  is the increment of weights }
1. INITIALIZE
     $\Delta_{ij}(t) = \Delta_0$  {initial step size}
REPEAT
{
2. Compute Gradient  $\frac{\partial E}{\partial w_{ij}}(t)$ 
    FOR all weights and biases
    {
3. IF  $(\frac{\partial E}{\partial w_{ij}}(t-1) * \frac{\partial E}{\partial w_{ij}}(t) > 0)$  THEN
        {  $\Delta_{ij}(t) = \min(\Delta_{ij}(t-1) * \eta^+, \Delta_{max})$  {delt_inc}
           $\Delta_{wij}(t) = -\text{sign}\left(\frac{\partial E}{\partial w_{ij}}(t)\right) * \Delta_{ij}(t)$ 
           $w_{ij}(t+1) = w_{ij}(t) + \Delta_{wij}(t)$ 
           $\frac{\partial E}{\partial w_{ij}}(t-1) = \frac{\partial E}{\partial w_{ij}}(t)$  }
4. ELSE IF  $(\frac{\partial E}{\partial w_{ij}}(t-1) * \frac{\partial E}{\partial w_{ij}}(t) < 0)$  THEN
        {  $\Delta_{ij}(t) = \max(\Delta_{ij}(t-1) * \eta^-, \Delta_{min})$  {delt_dec}
           $\frac{\partial E}{\partial w_{ij}}(t-1) = 0$  }
5. ELSE IF  $(\frac{\partial E}{\partial w_{ij}}(t-1) * \frac{\partial E}{\partial w_{ij}}(t) = 0)$  THEN
        {  $\Delta_{wij}(t) = -\text{sign}\left(\frac{\partial E}{\partial w_{ij}}(t)\right) * \Delta_{ij}(t)$ 
           $w_{ij}(t+1) = w_{ij}(t) + \Delta_{wij}(t)$ 
           $\frac{\partial E}{\partial w_{ij}}(t-1) = \frac{\partial E}{\partial w_{ij}}(t)$  }
    }
}
UNTIL (converged)

```

Figure 25: Pseudo-code of RP algorithm

The parameters used in training the neural network using RP algorithm are described in Table 4.

Table 4: Resilient Backpropagation training parameters

Parameter	Description
<i>epochs</i>	The training stops if number of iterations exceeds <i>epochs</i> parameter.
<i>goal</i>	Training stops if the performance function drops below <i>goal</i> parameter.
<i>min_grad</i>	Training stops if the magnitude of the gradient is less than <i>min_grad</i> parameter.
<i>max_fail</i>	Used for early stopping.
<i>delta0</i>	Initial step size.
<i>delt_inc</i>	The update for each weight and bias is increased by <i>delt_inc</i> whenever the derivative of the performance function with respect to that weight has the same sign for two successive iterations.
<i>delt_dec</i>	The update value is decreased by a factor <i>delt_dec</i> whenever the derivative with respect to that weight changes sign from the previous iteration.
<i>deltamax</i>	Maximum step size.
<i>time</i>	Training stops if training time is longer than <i>time</i> parameter (seconds).

3.4.3.3 Scaled Conjugate Gradient

As discussed previously Scaled Conjugate Gradient (SCG) [72] is used for both function approximation or pattern recognition tasks but in topologies with high number of weights. The algorithm is based upon a class of optimization techniques in numerical analysis known as the Conjugate Gradient Methods. SCG uses Hessian matrix of the network but requires only $X(N)$ memory usage, where N is the number of weights in the network. A pseudo-code to the SCG gradient is described in Figure 26:

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:

$$\sigma_k = \sigma / |p_k| \quad \{ p_k \text{ is non-zero weight vector from training set } \}$$

$$s_k = \frac{E'(w_k + \sigma_k p_k) - E'(w_k)}{\sigma_k} \quad \{ s_k \text{ is the approximation of the Hessian matrix } E''(w_k) \}$$

$$\delta_k = p_k^T s_k \quad \{ \delta_k \text{ is the error } \}$$
3. Scale s_k :

$$s_k = s_k + (\lambda_k - \bar{\lambda}_k) p_k \quad \{ \lambda \text{ regulates the indefiniteness of the Hessian matrix} \}$$

$$\delta_k = \delta_k + (\lambda_k - \bar{\lambda}_k) |p_k|^2$$
4. **IF** $\delta_k \leq 0$ **THEN** make the Hessian matrix positive define:

$$s_k = s_k + (\lambda_k - 2 \frac{\delta_k}{|p_k|^2}) p_k$$

$$\bar{\lambda}_k = 2(\lambda_k - \frac{\delta_k}{|p_k|^2})$$

$$\delta_k = -\delta_k + (\lambda_k) |p_k|^2, \quad \bar{\lambda}_k = \lambda_k$$
5. Calculate step size:

$$\mu_k = p_k^T r_k, \quad \alpha_k = \frac{\mu_k}{\delta_k}$$
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** $\Delta_k \geq 0$ **THEN** a successful reduction in error can be made:

$$w_{k+1} = w_k + \sigma_k p_k$$

$$r_{k+1} = -E'(w_{k+1})$$

$$\bar{\lambda}_k = 0, \text{ success} = \text{true}$$
 - 7a. **IF** $k \bmod N = 0$ **THEN** restart algorithm: $p_{k+1} = r_{k+1}$
ELSE create new conjugate direction:

$$\beta_k = \frac{|r_{k+1}|^2 - r_{k+1}^T r_{k+1}}{\mu_k}$$

$$p_{k+1} = r_{k+1} + \beta_k p_k$$
 - 7b. **IF** $\Delta_k \geq 0.75$ **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 26: Pseudo-code for SCG algorithm

The parameters used in training the neural network using SCG algorithm are described in

Table 5:

Table 5: Scaled Conjugate Gradient training parameters

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

After the NN was trained, next step is to verify the performance accuracy of the network.

3.5 Performance Reached (step 9)

Predictive accuracy refers to how well a given model accounts for examples that were not used in inducting the model. Performance functions include Root Mean Square, Mean Absolute and Mean Absolute Percent depicted in Table 6.

Table 6: Performance functions

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (t_i - a_i)^2} \text{ - Root Mean Square Error} \quad (19)$$

$$MAE = \frac{1}{N} \sum_{i=1}^N |t_i - a_i| \text{ - Mean Absolute Error} \quad (20)$$

$$MAPE = \frac{1}{N} \frac{\sum_{i=1}^N |t_i - a_i|}{a_i} \times 100 \text{ - Mean Absolute Percent Error} \quad (21)$$

Where e_i is the error of Output node i , t_i represents Targets and a_i Actual _Output, N is the number of examples used during testing process.

If the performance is unsatisfactory, then the topology of the network is changed. The number of the neurons in the hidden layer or the number of hidden layers is increased progressively until the performance is satisfactory. On the contrary, if the train performance is satisfied for function approximation or pattern recognition problem, the next step is to extract knowledge learned by the network.

3.6 TREPAN Module

The algorithm adopted in this work and used for extracting the knowledge embedded in trained neural networks is TREPAN [73] (TREes PARroting Networks). The algorithm builds a decision tree representing the function that a neural network has learnt (Figure 27). TREPAN is designed to extract a decision tree that maximize the tradeoff between fidelity and concision. First, it uses neural network's training set to construct models of the data distribution in the problem domain and uses these models to generate instances or membership queries. Secondly, 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 weights matrix, biases matrix, and the activation functions of the hidden and output layers. The algorithm also requires the input/output data which was used in training the NN. The output of TREPAN is a decision tree and is stored as an array of node structures. The topology of the tree is determined by assigning an index to each node, with the root node having index 1, and each node storing the indices of its parent and children. Binary splitting tests are used at the nodes, so that each node has either zero or two children. A node with no children is a leaf; a node with two children is a fork. The class assigned by the tree to examples reaching a node is determined by majority vote: the class with the largest number of training examples reaching the node wins.

Trepan differs from other decision tree algorithms in three significant areas. First, the order in which the nodes of the tree are expanded is based on a best-first method. Each node is assigned a priority, defined to be the proportion of examples misclassified by the node. The algorithm maintains a queue of leaf nodes, ordered by their priority, and successively expands the node at the head of the queue. Nodes with higher priorities are processed first, as they offer the greatest chance of increasing the number of correctly classified training examples. New child nodes are placed in the queue if they have non-

Chapter 3 Design and Development of a Knowledge Discovery System in Inventory Management

zero priority; i.e. some of the examples reaching the node are misclassified. Both the tree and the queue are initialized with the root node, which all the training examples pass through. The process stops when the number of nodes in the tree exceeds some pre-defined maximum size, or when the queue is empty.

INPUT: Oracle (), training set S , feature set F , *min_sample* parameter, stopping criteria

1. **FOR** each example $x \in S$
2. class label for $x := \text{Oracle}(x)$
3. initialize the root of the tree, R , as a leaf node
4. construct a model M of the distribution of instances covered by node R
5. $\text{query_instances}_R := \text{DRAWSAMPLE}(\{\}, \text{min_sample} - |S|, M)$
6. use S and query_instances_R to determine class label for R
7. initialize *Queue* with tuple $[R, S, \text{query_instances}_R, \{\}]$
8. **WHILE** *Queue* not empty and global stopping criteria not satisfied
9. remove [node N , S_N , query_instances_N , constraints_N] from head of *Queue*
10. $T := \text{CONSTRUCTTEST}(F, S_N \cup \text{query_instances}_N)$
11. make N an internal node with test T
12. **FOR** each outcome, t , of test T
13. make C , a new child node of N
14. $\text{constraints}_C := \text{constraints}_N \cup \{T=t\}$
15. $S_C := \text{members of } S_N \text{ with outcome } t \text{ of test } T$
16. construct a model M of the distribution of instances covered by node C
17. $\text{query_instances}_C := \text{DRAWSAMPLE}(\text{constraints}_C, \text{min_sample} - |S_C|, M)$
18. use S_C and query_instances_C to determine class label for C
19. **IF** local stopping criteria not satisfied **THEN**
20. put $[C, S_C, \text{query_instances}_C, \text{constraints}_C]$ in *Queue*

OUTPUT: tree with root R

Figure 27: Pseudo-code of TREPAN algorithm

Where *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.

Chapter 3 Design and Development of a Knowledge Discovery System in Inventory Management

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 for membership queries. The CONSTRUCTTEST function is used for 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*. The 2 of (*a1*, *a2*, *a3*) is logically equivalent to (*a1* and *a2*) or (*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 of 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. At each stage new tests are formed from the tests in the current beam and the list of Boolean splits: an *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*). To avoid over-fitting, the training data, tests are considered for inclusion in the beam only if the partitioning of the training set they produce 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 to expand 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 modeled by their empirical distributions, and kernel density estimates are

Chapter 3 Design and Development of a Knowledge Discovery System in Inventory Management

used to model continuous inputs. As each node is created, sufficient training cases are generated so that a pre-specified 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.

TREPAN module is divided in four steps including Attributes definition (step 10 of the framework from Figure 13), Options Setup (step 11), Data Definition (step 12) and Oracle Setup (step 13) which are described in next subchapters.

3.6.1 Attributes definition (step 10)

The attributes of the input and output variables for the oracle are defined in a text file (Table 7). The number of attributes is the same with the number of inputs into neural network,

Table 7: Attributes definition

Name	type	[values]
x	I or R or N	If x is N then values: x1, x2, x3...

“Name” is the variable name and type is N (nominal), I (integer), or R (real). Nominal variables are followed by a list of their values (as strings). The final line defines the output variable, which must be nominal with two values.

An example is created with the purpose to mediate the understanding of the algorithm. Therefore, in Figure 28 there are seven attributes defined: *cs*, *nc*, *lt*, *as*, *s*, *it* and *class* (representing in order: customer satisfaction, number of customers, lead time, actual safety stock, seasonal factor, inventory turnover). Each one has its own type (N –nominal, I- integer, and R –real). Real attributes can take values between $(-\infty, +\infty)$ but are usually scaled between $[-1, 1]$ or $[0, 1]$ in transformation step. On the other hand for nominal attributes must specify a finite number of elements, e.g. the customer satisfaction level can

Chapter 3 Design and Development of a Knowledge Discovery System in Inventory Management

denoted with three values: 1, 2, 3. Value 1 means a customer is unsatisfied, 2 –satisfied and 3- excited. Therefore, the number of customers (nc) are integer values (denoted by *I* in attributes table). Lead time (lt) has nominal values (N) with four values: 2,3,4,5 – representing the number of weeks. Actual safety stock (as) has real values (R). Seasonal factor (s) has two nominal values: 0 and 1. With 0 value is coded the absence of seasonal factor and by 1 value as its presence. Inventory turnover (it) is represented with real values. Lastly, the attribute *class* is always nominal (N) and always has two values. In our example the attribute *class* has two values: *down* and *up*. With *down* is denoted the next level of inventory if is decreasing with respect to previous level and with *up* value if next inventory level is increasing.

After Attributes Definition step in which input variables has been defined and are in fact the Inputs used in training the NN, next step is to setup TREPAN Options (Table 8). At this step the efficiency and output of the algorithm can be controlled for example by limiting the maxim tree size or selecting the number of M-of-N tests made at each node. A description of all the variables used to setup the algorithm is mare in next section.

Figure 28: Example of attributes definition in TREPAN

3.6.2 Options Setup (step 11)

Table 8 depicts the options setup variables used by TREPAN algorithm.

Table 8: Options used to control TREPAN algorithm

Variable name	Values	Description
verbose	Controls the level of information produced as the program runs.	
	0	display only error messages
	1	minimal progress messages
	2	include details of the beam search
	3	include details of the variable distributions at the nodes
random_seed	Sets the seed for the uniform (rand) and normal (randn) random number generators.	
	0	set seed from system clock
	non 0	set seed to specified value
max_tree_size	10	The maximum size of tree to grow (10 in our case)
max_m_of_n	2	The maximum number of binary splits in an m -of- n test. The default value of 2 means the m -of- n tests are equivalent to logical ANDs and ORs.
min_sample_size	1000	Examples are generated until this number reach each node.
beam_width	2	The width of the beam used to search for an optimal m -of- n test.
min_objects	20	Nodes reached by less than this number of training cases are not expanded.
max_splits	100	Real input variables often produce very many candidate splits. The candidate splits at each node are ordered by information gain, and only the top max_splits are used during each beam search.
mofn_alpha	0.05	The significance level used when comparing m -of- n tests during the beam search.
dist_alpha	0.1	The significance level used when comparing distributions to see if they differ.

3.6.3 Data Definition (step 12)

At step 12, data necessary for training and pruning trees is defined. Each set is stored in whitespace-separated files with one line per case: *name val1 val2*, here *name* is the case name and *val1*, *val2*, etc., are the values of the variables for that case. The variables must appear in the order specified in the attributes file. For example Figure 29 shows data table feed as input into TREPAN. First column (*name*) is omitted by algorithm during of attributes reading. In Figure 29, *cs* stands for customer satisfaction (possible nominal values 1,2 or 3), *nc*- number of customers (integer values), *lt* – lead time (nominal values 2,3,4 or 5), *as* – actual safety stock (real values), *s* – seasonal factor (nominal values 0 or 1), *it* – inventory turnover (real values scaled between [0,1]) and *class* –which can be *up* or *down*. For example a vector used in one iteration by TREPAN algorithm is composed by case $p-0 = (2 \ 17 \ 2 \ 212 \ 0 \ 0.6003 \ up)$. This can translated by the algorithm: For case number $p-0$, WHEN $cs=2$, $nc=17$, $lt=2$, $as=212$, $s=0$ and $it=0.6003$ *class* is *up*. This procedure is repeated for all cases existed in the data definition table.

name	cs	nc	lt	as	s	it	class
p-0	2	17	2	212	0	0.60034914	up
p-1	1	33	5	373	0	0.913075165	up
p-2	1	39	3	271	1	0.805093965	up
p-3	3	66	3	330	1	0.331382739	down
p-4	3	42	2	224	0	0.586313711	up
p-5	1	59	4	466	0	0.668883094	up
p-6	3	19	4	341	0	0.074246605	up
p-7	3	37	2	363	1	0.585151218	down
p-8	3	39	2	235	1	0.48750726	down
p-9	3	51	4	444	1	0.223913934	down

Figure 29: Example of data definition table

After the Data was defined next step is to create the Oracle, the function which models the trained NN.

3.6.4 Oracle Setup (step 13)

At this step the oracle from which to extract the tree is created. This function takes as input a matrix where each row contains the input variables for a single case and returns a column vector containing the class assigned to each case by the oracle.

Chapter 3 Design and Development of a Knowledge Discovery System in Inventory Management

To avoid imposing an artificial ordering, a 1-of-N encoding is made to the input attributes during the NN training. For example in Figure 30 the attribute *cs* has three possible nominal values (1, 2 and 3). In order to code the values an identity matrix is used.

$$I_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \text{ where first value 1 is assign first line of the matrix, value 2 second line}$$

and last value 3 third line of the matrix. Therefore after coding, value 1 becomes “1 0 0”, value 2 becomes “0 1 0”, and value 3 “0 0 1”. Same procedure applies for all nominal values. As a conclusion each nominal attributes creates N inputs in neural network input, where N is number of elements of the corresponding attribute.

After the creation of the oracle function, TREPAN algorithm can be used. In our example (developed through Step 10, 11 and 12), one possible tree extracted by the algorithm is represented in Figure 30.

Figure 30: Example of interpreted decision tree extracted by TREPAN

Chapter 3 Design and Development of a Knowledge Discovery System in Inventory Management

There are three nodes, depicted with Node 1, 2 and 3. Node 1 is named the *root* of the tree. Node 2 and 3 are the children of Node 1. Because Node 3 has no children, it is considered as *leaf*.

In Table 9 the architecture of a single node is depicted. For more information on trees extracted with Trepan please see Appendix 3.

Table 9: TREPAN Tree interpretation

node	The index of the node (first node of tree is named root)
parent	The index of the parent node
child1	The index of the first child node
child2	The index of the second child node
leaf	Whether the node is a leaf (the node has no children is a leaf)
pruned	Whether the node has been pruned (yes/no).
#class 1	The number of examples in class one reaching the node (the number of examples in class one in the original training set is in parentheses)
#class 2	The number of examples in class two reaching the node (the number of examples in class two in the original training set is in parentheses)
class	The class assigned to the node (the winner of the previous two entries)
priority	The priority assigned to the node (priority decreases from the root to bottom)
gain	The information gain of the m-of-n test at the node
m	The value of m in the m-of-n test (e.g. m=2 in “2 of {cs=1, nc<=50, s=1}” THEN ...)
split	Split values N (e.g. cs=1, nc<=50, s=1), in this case N=3.

The output of TREPAN is a decision tree which has to be inspected by the user in term of the accuracy, fidelity, consistency and comprehensibility.

3.7 Knowledge assessment (step 14)

Knowledge assessment should be made by taking the following into considerations:

1. Accuracy
2. Fidelity
3. Consistency
4. Comprehensibility

A decision tree is considered to be accurate if it can correctly classify previously unseen examples. Similarly a decision tree is considered to display a high level of fidelity if it can interpret the behavior of the neural network from by capturing all of the information embodied in the NN. An extracted decision tree is considered to be consistent if, under differing training sessions, the NN generates decision trees which produce the same classifications of unseen examples. Comprehensibility refers to how easily one can inspect and understand a decision tree constructed by TREPAN algorithm. Sometimes the learning method which constructs the model with the best predictive accuracy is not the method that produces the most comprehensible model. Trees obtained using TREPAN algorithm are very easily interpreted, therefore it has high comprehensibility (Figure 30).

If the obtained tree has a high accuracy (it can correctly classify a test set which was unused during the training process), the knowledge depicted by tree can be used to understand the relationship between the input variables. As a result, it is easy to identify which input is more important in determining the class created.

3.8 Use of knowledge (step 15)

Knowledge is the output of the proposed framework. At this step , knowledge in the form of decision tree, extracted from trained network by TREPAN algorithm is used to explain the NN reasoning in found relationships from inputted data. In case of causal models, in which at the input are presented multiple independent variables, the rules can identify and explain significance of the most important variable (Figure 31). As discussed in Literature Review chapter, inventory management performance is influenced by many factors (e.g. lead time, customer demand etc.). These factors can be quantified and used to train a neural network with the purpose to predict the level of inventory level. For

Chapter 3 Design and Development of a Knowledge Discovery System in Inventory Management

inventory management managers it would be desirable also to know which of the inputted factors are more relevant and how much influences output (inventory level). It can be stated that NN behavior is interpreted by the TREPAN algorithm. Therefore, managers can act according to the knowledge represented in the form of decision rules and adjust the policy of the inventory management. For example, from decision tree extracted by Trepan in Figure 30 we can conclude that if lead time is high (five weeks) or actual safety stock is less than 300 units then next inventory level is going to decrease in comparison to the current one.

Figure 31: Use of knowledge in Inventory Management

To sum up, this chapter presented an algorithm to extract knowledge in a form of decision trees from Inventory databases. The proposed system named Inventory Knowledge Discovery System (IKDS) requires to clean and transform before the data is entered Neural Network Module and TREPAN Module. Otherwise, the performance of the trained NN will be affected. NN training and TREPAN algorithm are coded in MatLab 2007b. With the function of TREPAN, decision tree is built for supporting decision making and knowledge can be extracted. The knowledge extracted in the form of decision tree is stored in two formats: a Matlab binary format used to make predictions with new data, and a text format used to examine and understand the tree.

CHAPTER 4. EVALUATION OF THE INVENTORY KNOWLEDGE DISCOVERY SYSTEM

In this chapter, the Inventory Knowledge Discovery System (IKDS) from previous chapter is tested by two cases:

- 1) A time series forecasting of the demand.

Data is recorded over a period of three years and represents the demand of one item carried by an Australian subsidiary of a Japanese car company, and second case

- 2) An analysis of the impact of demand, lead time and cycle service level on safety stock. Data represents inventory level of Dell Company which sells PCs.

4.1 Time Series Forecasting of the demand

First case is a time series forecasting problem. Table 10 showed that data used was recorded over a period of three years and represents the demand of one item carried by an Australian subsidiary of a Japanese car company [74].

Table 10: 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

The purpose is to train a NARX network to forecast demand on month 36 (Figure 32). Therefore first 35 values are used for training the network.

Figure 32 Distribution of demand over time

Chapter 4 Evaluation of the Inventory Knowledge Discovery System

Training of the NARX network is made using first three values of the time series data as inputs and the fourth as the targets. Considering a time series data $t_1, t_2, t_3, t_4, \dots, t_n$ with n values (in our case $n=35$ months) first three values t_1, t_2, t_3 are used for training and next value t_4 is used as target. For the second training set t_2, t_3, t_4 are used for training and t_5 is used as a target. The procedure is repeated for $n-3$ months. Therefore the NARX network has four inputs (considering the recurrent input also) and one output. A sample of the data used for training is depicted in Appendix 1. Four NARX network are trained using 10, 15, 20 and 25 neurons in hidden layer. In Figure 33, it can be observed that best training performance is obtained for 20 neurons topology. After the network is trained, the next step is to forecast for the demand of the 36th month. Using the previous three values from the time series (57; 23; 39) as input, the output of the network is 23. 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 11.

Table 11: RP training parameters for 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

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

Chapter 4 Evaluation of the Inventory Knowledge Discovery System

Figure 33: Performance assessment for four NARX topologies (best validation performance obtained for 20 hidden neurons topology)

In Table 12, NARX network accuracy is compared with ARIMA (Appendix 1), Moving Average MA(3) and Moving Average MA(2). It can be observed that best accuracy in forecasting demand for the 36th month is obtained by NARX network (Figure 34).

Table 12: 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

Figure 34: Errors comparisons

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 time series and is calculated [75] as:

$$\alpha_i = \frac{y(t_{i+1}) - y(t_i)}{t_{i+1} - t_i} \quad (23)$$

Next, slopes are coded like in Figure 35. **IF** $\alpha_i \geq 0$ **THEN** $\alpha_i = 1$, **ELSE** $\alpha_i = 2$. If slope is positive or equal with zero then is encoded as "1". If slope is negative then is coded as "2". α_i denotes the coded value of the slope α_i .

Figure 35: Proposed coding of slopes for time series case

Chapter 4 Evaluation of the Inventory Knowledge Discovery System

The idea is to use first three values (t_1 , t_2 , t_3) and their slopes (α_1 , α_2 , α_3) to predict the 4th slope if is positive or equal to zero (coded as 1) or is negative (coded as 2). Note that α_2 is the slope calculated between value t_2 and t_3 , while α_3 is the slope calculated between t_1 and t_3 (Figure 36).

Figure 36: Slopes representation for each value of demand

The variables used to feed the NARX network are depicted in Table 13. A sample used to train the NN is depicted in Appendix 2.

Table 13: Variable selection for time series case

Variables	Abbreviation	Type	Range of values
Level t1	l1	R (real)	21-97
Slope1	alpha1	N (nominal)	1,2
Level t2	l2	R	21-97
Slope2	alpha2	N	1,2
Level t3	l3	R	21-97
Slope3	alpha3	N	1,2
Class		N	down/up

Tree extracted using TREPAN is described in Figure 37 (Appendix 3 shows further information on decision tree obtained). Use Node 1 of Figure 37 as illustration.

Chapter 4 Evaluation of the Inventory Knowledge Discovery System

If both of the following condition are satisfied

- (1) $\alpha_3=1$ (i.e. slope 3 is positive or zero) and
- (2) $I_3>56$ (i.e. demand a Level t_3 shown in Figure 36 is more than 56)

Then it is predicted that the demand will increase.

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 are decreased, the demand will decrease.

Figure 37: Tree extracted by TREPAN for time series case

4.2 Impact of demand, lead time and cycle service level on safety inventory

For the second experiment, inventory level of Dell Company [25] which sells PCs is analyzed with the purpose to determine the impact of demand, lead time and cycle service level on safety stock. 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 safety stock (SS) is the inventory carried out to improve product availability in the presence of uncertainty in both demand and supply. Therefore, the appropriate level of safety stock is calculated with respect to uncertainty of demand and supply, and desired level of product availability. If the uncertainty of demand and supply

Chapter 4 Evaluation of the Inventory Knowledge Discovery System

increases, the required level of safety stock increases also. If the preferred level of product availability increases, the required level of safety stock also increases (Figure 38).

Figure 38: Relation between Service Level and Safety Factor

Let's consider the situation of a continuous review system (Figure 39). The level of Safety stock at time period t_i is calculated [25] as: $SS_i = \text{norminv}(\text{CSL}) \sigma_{dl_i}$, where $\text{norminv}()$ is the normal inverse cumulative distribution function, Cycle Service Level (CSL) has the range between [0.5 to 0.99], and σ_{dl_i} is standard deviation of demand during lead time. The formulas to calculate the variables involved in calculating safety stock at time t_i are described below:

$$\text{Standard deviation of demand during lead time } \sigma_{dl_i} = (L_i \sigma_{d_i}^2 + D_i^2 \sigma_{l_i}^2)^{\frac{1}{2}} \quad (24)$$

$$\text{Average demand, } D_i = \frac{1}{N} \sum_{i=1}^N d_i \quad (25)$$

$$\text{Standard deviation of demand per period, } \sigma_{d_i} = \left(\frac{1}{N-1} \sum_{i=1}^N \left(d_i - \frac{1}{N} \sum_{i=1}^N d_i \right)^2 \right)^{\frac{1}{2}} \quad (26)$$

$$\text{Average lead time for replenishment, } L_i = \frac{1}{N} \sum_{i=1}^N l_i \quad (27)$$

$$\text{Standard deviation of lead time, } \sigma_{l_i} = \left(\frac{1}{N-1} \sum_{i=1}^N \left(l_i - \frac{1}{N} \sum_{i=1}^N l_i \right)^2 \right)^{\frac{1}{2}} \quad (28)$$

Chapter 4 Evaluation of the Inventory Knowledge Discovery System

Figure 39: Continuous review.

Inventory status is continuously checked and an order is placed when the inventory declines to the reorder point (ROP), T_i is the time between successive orders.

Demand on time period t for PCs at Dell is normally distributed and using above equations, the variables used to calculate the safety stock level are depicted in Table 14:

Table 14: 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
std dev 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
std dev 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	44913	9947	24660
cycle service level		0.657879	0.65275	0.58715	0.5831	0.5288	0.96816	0.6694	0.84639	0.9664

Variables used to train the TREPAN algorithm are described in Table 15.

Table 15: 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)

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

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

In other words **IF** safety stock at time period t_i is greater or equal with previous safety stock (period t_{i-1}) then the class assigned is “increase” (the safety stock has increased) **ELSE** class assigned is “decrease” (the safety stock has decrease).

The coding in Matlab to generate the numbers, train the network and to construct the oracle is described in Appendix 4.

This is a pattern recognition problem with an increased number of inputs and weights, therefore training algorithm is Scaled Conjugate Algorithm (SCG) and a Feedforward NN is selected as architecture. From 250 samples, 60% are used for training, 20% for validation and last 20% for testing. After training 4 separate feedforward NN networks with 15,20,25,30 neurons in the hidden layer, the topology with 20 neurons obtained best results in fitting the data. Best validation performance is 0.233 at epoch 60 (Figure 40).

In Figure 40, with blue color is depicted the training performance of the training set (60% from the total 250 samples), with green color is depicted the performance of the

Chapter 4 Evaluation of the Inventory Knowledge Discovery System

validation set (20%) and with red color the performance of the test set (20%). Training stops when validation set is optimal (goal is 0).

- 10 hidden neurons

- 15 hidden neurons

- 30 hidden neurons

- 20 hidden neurons

Figure 40: Training performance achieved by tested network topologies (best performance achieved by 20 hidden neurons topology)

Chapter 4 Evaluation of the Inventory Knowledge Discovery System

Regression analysis (Figure 41) 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 neural network selected in this simulation (20 neurons topology) performs satisfactory with values around 0.9 for training sample, 0.87 for validation set and 0.874 for test samples.

Figure 41: Regression analysis for 20 neurons topology selected

Chapter 4 Evaluation of the Inventory Knowledge Discovery System

After training the network the TREPAN algorithm is used to extract learnt relationships by the network. Input data are used to train the NN are depicted in Appendix 4. An interpretation of the extracted decision tree is made in Figure 42 (Appendix 5).

Figure 42: Tree extracted by TREPAN

4.3 Discussion and conclusions

This chapter has verified the efficiency of the Inventory Knowledge Discovery System (IKDS- Figure 13) in two cases: first is a time series forecasting of demand of one item carried by an Australian subsidiary of a Japanese automobile company and the second case is the analysis of the impact of demand, lead time and cycle service level on safety stock.

Chapter 4 Evaluation of the Inventory Knowledge Discovery System

In the first case related to time series forecasting, it can be concluded that NARX network have superior performance in comparison with traditional forecasting methods (ARIMA or MA). TREPAN algorithm can be used to extract knowledge embedded in NARX network to explain the functionality of the network. The interpretation of the tree extracted in causal forecasting case can be done as: IF both slope 3 is positive and l3 (level t3) 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 slope3 and slope1 are negative then next slope is positive else the slope is negative. The rules covering high fidelity with almost all the time series data used in training process. An inventory manager could use this knowledge in the form of rules to manage inventory more efficiently.

The second case has the purpose to understand the available data, more exactly to determine the impact of demand, lead time and cycle service level on safety stock calculation at Dell Company. Therefore a Feedforward NN is trained with the purpose to learn the relations between input variables (average demand, standard deviation of demand, average lead time, standard deviation of lead time, previous safety stock and cycle service level) and output which is “1” if the calculated safety stock at a period t of time has increased in comparison to the previous safety stock (at time $t-1$), or “-1” if it has decrease. Interpreting the decision tree obtained using TREPAN algorithm can be done as: For node 1, IF cycle_service_level is less than 0.70 then the next safety stock level will decrease in comparison to the actual one (node 2), ELSE for all cycle_service_level ≥ 0.7 , IF previous_safety_stock (node 3) is less than 8544 units then the next safety stock will increase also, ELSE for all cycle_service_level ≥ 0.7 and previous_safety_stock ≥ 8544 , IF mean_demand < 2086 then the next safety stock will increase, ELSE next safety stock will decrease.

CHAPTER 5. CASE STUDY

This chapter strives to test the framework proposed in Chapter 3 (Figure 13) to an inventory problem encountered at Panasonic Refrigeration Branch located in Singapore. The Company lacks in a suitable forecasting technique for short periods of time. A NARX network is chose to forecast with promising results. Next TREPAN algorithm is used to extract a decision tree which could be used by managers to improve inventory level.

Panasonic Refrigeration Devices in Singapore (PRD) is a division of MEI Co, which is empowered with the production of refrigeration compressors. The company produces four types of compressors (Inverter, FNQ, ES, S) but each compressor can be customized at the request of each customer (Mitsubishi, LG, Bosch, Haier etc). For example, if one compressor is usually assembled from around 100 standard components, mass customization of the compressor can be up to 200 interchangeable. Informational flow encountered at PRD is depicted in Figure 43:

Figure 43: Information Flow in PRD

Chapter 5 Case Study

1) Sales Department (in Panasonic case is integrated with Customer Service) receives the orders from the customers (Bosch, LG, Mitsubishi, Haier and even Panasonic) and passes it to Production Control (PC) which will decompose the number of compressors (105 as in example) to the elementary parts. Then PC will settle the schedule of manufacturing and the number of raw materials necessary to production lines. Module number 105 represents the total number of compressors received by sales department; Number 10528 represents the components necessary to assembly line to create the 105 compressors.

2) In the second step, Information System department will print forms received from PC and pass them to Inventory Department which based on the last month inventory calculates the number of components necessary to supply manufacturing and to fulfill company buffer (This value depends on the location of suppliers - Table 16).

Table 16: Lead Times from suppliers

Supplier	Local	Neighbor	Overseas
	Daily (3 times a day)	1 Month (few weeks)	1-4 Months
Time	JIT	VMI	VMI

3) In the third step the Inventory department passes an order to Purchasing Order Department, which will issue a Purchase order form to Suppliers (Figure 44).

Chapter 5 Case Study

Figure 44: PRD purchasing process

By the year 2003 the company started a new policy to achieve global excellence in the year 2010, using two strategies in inventory management, JIT and VMI.

JIT is to minimize inventory, improve product quality, maximize production efficiency and provide highest customer service levels. It is basically a philosophy of doing business. In PRD, JIT is used with success in supplying raw materials to manufacturing (steel plate). Main benefit of JIT is the decreased warehouse space, no inventory need and improved response time. For example, steel plate from which is made upper and lower hull of compressor can occupy an increased space in warehouse. IT means no stock buffer. Without buffer, it may have adverse effect on the production line. The production line can be stopped if the steel plates are absent. To ensure JIT to work efficiently, the suppliers have to be very reliable. In PRDS, case the steel plate supplier has to deliver three times a day raw materials to manufacturing. To do it in an efficient way, suppliers' locations have to be very close to PRD.

On the other hand Vendor Managed Inventory (VMI) implies that the buyer of a product (in our case PRD) provides certain information to a supplier of that product and the supplier takes full responsibility for maintaining an agreed inventory of the material.

Chapter 5 Case Study

Some features pertaining to VMI include:

- Constantly communication of inventory, planned promotions and stock-outs.
- Trucks are filled in a prioritized order. For example, items that are expected to stock out have top priority, then items that are furthest below targeted stock levels, and finally items that are least above targeted stock levels.

Main problem in implementing VMI is that not all of Panasonic suppliers agree to VMI strategy, as a result, accurate prediction about safety stock and tight inventory control are still carried by Panasonic.

5.1 Problem Definition

1) The first problem encountered at Panasonic Refrigeration Devices Company was the necessity of a suitable forecasting technique used for short periods of time, more exactly two weeks. Even though the Sales Department provides the information necessary to produce the new quantity of compressors, it seems that this information is not stable over time, modifying from 90% accuracy with short time span to 60% for the mid time span and there are about 30% for long time span (Figure 45).

Figure 45: Accuracy of forecasting in time

2) A second problem found at PRD Company is related with inventory management process. The inventory data is not integrated to the central system and not updated in real time. There are three managers who are responsible with inventory process. The changes made by one of them is not in real-time updated into the system; hence the others may work with un-updated information.

Chapter 5 Case Study

3) In order to maintain the competition edge or having the advantages over other companies, PRD Company provides the flexibility for the customers to change or cancel an order even though an order already is in production. This can result in time and money lost, because the manufacturing schedule must be revised, and in case of cancellation, surplus compressors are stored in warehouse.

Next subchapter describes the ordering policies used by the company.

5.2 Ordering Methods

In Panasonic Company case we identified three types of ordering methods:

1) *Monthly Ordering Method* Suppliers are mainly from overseas. The approach is described in Figure 46 and the steps are shown as following.

- Lead time ranging from 1 month to 4 months depends on commodities.
- First issue is based on 1/3 of monthly requirements
- First revision is based on 1/3 of monthly requirements
- Second revision is based on 1/3 of monthly requirements
- Items involved are flapper steel, steel wire, glass mat, rubber parts.

Figure 46: Monthly ordering

B stands from Beginning; M from Middle and End period while ETA is Estimated Time of Arrival.

Chapter 5 Case Study

2) *Weekly Ordering Method.*

Weekly ordering method is adopted for local and Malaysia vendors. The approach is described in Figure 47 and the steps are shown as following

- Delivery frequency is 3 time per week (Malaysia only).
- Delivery frequency is everyday (Local).
- Items involved are Sintered Mat, A1 Parts, and springs

Figure 47: Weekly Ordering Method

3) *Daily Delivery Ordering Method*

The approach of Daily Delivery Ordering Method is described in Figure 48:

- Daily delivery ordering is based on PC fortnightly plan issued on 3rd working day
- Each cycle consists of 3 PO issue periods

Chapter 5 Case Study

- The 3 PO issue periods covers the 1st issue, 1st revision and 2nd revision.
- Items involved are magnet wire, shell blank, cast material

DAY

Figure 48: Daily Delivery Ordering Method

5.3 Safety Stock at PRD

By analyzing two samples of data from Panasonic RDS database (Figure 49), it can be easily identified that safety stock is set very high. Items analyzed are Hot Rolled Coil SQ-0230991 (left) and Hot Rolled Coil SQ-0231049 (right) used for shell production (both items are in R-MAT MRP excel table from CD).

Figure 49: Inventory, Production and Ordering for two items

Chapter 5 Case Study

Table 17 is a sample collected from Panasonic inventory database. The item is Hot Rolled Coil (SQ-0230991) overseas supply. For a more detailed sample database please check Appendix 6.

Table 17: Sample from PRD database

Item	Mth end Inv.	Sub-con. Inventory	May			Jun			Jul			Aug		
			Order	Prodn	Inv.	Order	Prodn	Inv.	Order	Prodn	Inv.	Order	Prodn	Inv.
SQ-0230991 (275)		C 31138	155365	18690 - 2/5		104685			113819			79142		
Hot Rolled Coil		B 84240	40000	(46643)					B 40000	(47914)		B		
2.3 x 991 SPHD		P 6890	18120	(46102) - 2/5								U		
1.247kg/pc			74575	(46644/46106) - 2/5										
(1.8 mth stock)			28532	226431		39481	186950		45703	181247		37460	143787	
2/5	99070		92460	100000	5.74	100000	100000	4.09	100000	100000	4.84	300000	300000	4.00
U/SHELL	(46645C-P080122M)	23/5-(47227)	(P080225M-PRDM)	(47918)	(P080319M-PRDM)	(48590)	(P080422M-PRDM)		(P080516M-PRDM)					

Formulas used by managers to calculate Inventory level for the next month are described below:

$$I_i = I_{i-1} + D_i - P_i$$

$$O_i = I_i - 1.8 * P_{i+1}$$

Where I_i is inventory at period i , D_i is the quantity delivered by the suppliers at period i , P_i is the quantity necessary for production of that item, and Q_i is the order places to the suppliers on i th month.

Current lead time ranging from 1 month to 4 month and depends on suppliers commodities. The constant 1.8 constraints inventory to 1.8 month stock.

5.4 Inventory Level Forecasting at PRD

Considering the steps from framework proposed in Chapter 3 (Figure 13) inventory records available on 5 months for item Hot Rolled Coil SQ-0231049 (see Appendix 6) are analyzed:

1) Goal definition. The goal is to design and train an Artificial Neural Network which best fits data available of above specified item with the purpose to forecast based on past records.

2) Data acquisition. This step is already done because the data to be analyzed have been retrieved from PRD database (Appendix 6). The Item is Hot Rolled Coil SQ-0231049 and the time series records are stored in excel table format.

Chapter 5 Case Study

3) Data selection. Data selection step extracts from the table (Appendix 6) only the records of Item SQ-0231049. After this step, the data is stored like in Table 18:

Table 18 :Monthly Inventory levels

Month	1	2	3	4	5
Inventory Level	211799	179617	263225	208790	185151

The relation between months and inventory level quantity is described in Figure 50:

Figure 50 :Inv. Level for item Hot Rolled Coil SQ -0231049

4) Data pre-processing. This step deals with data cleaning or enrichment. The records don't contain any missing values. As we have only five records of the past inventory levels, we need to enrich the data. Using an interpolation function (green line), we obtain the following chart (Figure 51):

Figure 51: Interpolation function

Dividing each month in 4 weeks and rounding up the last two decimals, we obtain table below:

Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Inv Lv	2118	1939	1842	1804	1796	1921	2229	2504	2632	2551	2386	2200	2086	2036	1992	1962	1952

Now we have 17 samples corresponding to 17 weeks. The goal is to predict at 9th week and 13th week (see Figure 52 - yellow colored dots).

Chapter 5 Case Study

P1 corresponds to actual value 2632 (9th week) and P2 to value 2086 (13th week).

Figure 52: Inventory level records chart

The values to be forecasted are at week 9, where the inventory level is increasing, and week 13, where the inventory level is decreasing.

5) Data transformation. This step changes data into a form suitable for the inputs into NN. Our data contains no categorical data (true/false) and the values are numerical, therefore no transformations are required.

6) Neural Network architecture selection.

We have the following two cases (6.1 and 6.2) corresponding to week 9 and week 13. We have chosen these two weeks because the trend is increasing in first case (6.1) and is decreasing for second case (6.2).

Case 6.1: For the first point to be predicted (2632 –actual value), we use a FFNN and a NARX network to forecast for 9th week. The data used for input to train, validate and test the data is depicted in Table 19:

Chapter 5 Case Study

Table 19: Training sample for P1

	Training set					TEST
Input	2118	1939	1842	1804	1796	1921
	1939	1842	1804	1796	1921	2229
	1842	1804	1796	1921	2229	2504
Output						
	1804	1796	1921	2229	2504	2632

Therefore, we have three inputs and one output while Levenberg-Marquardt algorithm is selected for training. The application randomly divides input vectors and target vectors into three sets as follows:

- 60% are used for training.
- 20% are used to validate that the network is generalizing and to stop training before overfitting.
- The last 20% are used as a completely independent test of network generalization.

After training the network the performance can be evaluated using performance function.

The below charts (Figure 53) represent the performance analysis for training (blue), validation (green) and test (red) samples:

FFNN P1 (2086)

NARX P1 (2086)

Figure 53: NN Performance analyses

Even though the training set is small (17 samples), both NN are able to learn efficiently the pattern of the data, and NARX network corresponding to point P1 (NARX P1) depicts a better performance in comparison with FFNN for P1.

Chapter 5 Case Study

Case 6.2: For the second point (2086-actual value), FFNN and NARX network are trained using Table 20:

Table 20: Training sample for P2

Training set										
Input	2118	1939	1842	1804	1796	1921	2229	2504	2632	2551
	1939	1842	1804	1796	1921	2229	2504	2632	2551	2386
	1842	1804	1796	1921	2229	2504	2632	2551	2386	2200
Output	1804	1796	1921	2229	2504	2632	2551	2386	2200	2086

Chapter 5 Case Study

Error represents the difference between the predicted value an week 13 and the actual value for P 2 which is 2086 (Figure 55).

Figure 55: Graphical representation of Error for P2

Applying the TREPAN algorithm for NARX network using the methodology described in Time Series forecasting (4.4.1), the following decision tree is obtained (Figure 56).

Figure 56: Tree extracted for Case Study

In other words, the tree can be interpreted as: If both slopes (alpha1 and alpha2 are “down” then next slope is “down” also ELSE IF both slopes are “up” then next slope is “up” also. In first case (6.1), the best result is obtained using a NARX NN. The difference

Chapter 5 Case Study

between actual value and predicted value for week 9 is 148. For the second case (6.2), the best result is obtained by NARX NN model. The difference between actual and predicted value is only 7. After the NN was trained to forecast, TREPAN algorithm is used to interpret the knowledge acquired by NN. Therefore, the manager can roughly have an idea whether they should adjust to increase or decrease the inventory based on the existing scenario.

5.5 Discussions

One of the solutions that PRD could be done to improve inventory management efficiency is to integrate JIT II which applies JIT concepts to the purchasing function by having a representative of the supplier locate at the buyer organization's facility. This approach assumes mutual understanding between the buyer and supplier. It helps improve suppliers' responsiveness, and creates a positive working environment.

For solving Problems 1, accurate forecasting method can be investigated. As accuracy of order information is higher and more data is recorded, NARX network can be used for time series forecasting with good results as it was shown previous in section 5.4.

For solving problem 2, synchronized and updated data is required. Data need to be stored in centralized database which is the initial step of the proposed system.

For solving problem 3, detecting the order change with some variables - economic downturn (A) and low unemployment rate (N), acquires the knowledge related to order change, order cancel pattern and formulate knowledge as rules. A neural network can be used to find the relations between input data and the TREPAN algorithm can explain the functionality of the network and the significance of the input variables. Therefore, once the managers know those factors, they can estimate the probability of order cancelation and order change based on different scenario.

CHAPTER 6. CONCLUSIONS AND FUTURE WORK

6.1 Summary of the findings

This work has made some new contributions in the application of advanced data analysis techniques to inventory management. Inventory Knowledge Discovery System (IKDS) was developed with the purpose to perform two tasks: IKDS helps to (1) forecast demand in inventory management and (2) get a better understanding of the variables involved in forecasting process. First task is achieved by using artificial neural networks (NN) while the second task is achieved by using TREPAN algorithm which extracts the knowledge learnt by trained NN in the form of decision trees. Extracting the knowledge embedded in NN weights could help in understanding the relevance of input parameters and the factors which are detrimental to inventory management. Decision rules embodied in the decision tree generated by TREPAN algorithm helps human user in decision making.

The performance of the proposed system (IKDS) has been evaluated and verified carefully in two situations. One is a time series forecasting problem for demand, another is related to leading time and cycle service level on warehouse safety stock. A real industrial application in Panasonic Refrigeration Devices has been included to illustrate the efficiency of the proposed system in industries.

In conclusion, knowledge is an important asset to the companies. Apart from working procedures, physical asset, and capital, knowledge plays a decisive role for effective production and inventory management. The quest to determine and understand the importance and relevance of factors, which influences the performance of forecasting in inventory management, represents an important task of knowledge discovery process.

6.2 Contribution of the research

The proposed methodology is a step towards the development of Inventory Knowledge Discovery Systems for extracting the knowledge and hidden pattern from the inventory data. The results show that forecasting with ANN is promising superior accuracy in comparison to traditional methods like Moving Average (MA) or ARIMA. Apart from

Chapter 6 Conclusions and Future Work

adopting neural networks to increase the accuracy of the forecasting result, this report has contributed to provide explanations of models and patterns learnt by the NN with TREPAN algorithm. Instead of determining the exact solution of inventory level and inventory position, this study may be the pioneer to adopt knowledge discovery techniques to analyze the reason behind the trend of inventory level. Through studying the slope of the inventory trend, industrial practitioner can make the decision to adjust the inventory so as to responsive to the dynamic business environment.

6.3 Limitations and Future Works

The limitation of the proposed system is that Trepan cannot describe the real-valued output so as to perform regression tasks. As one of the further works is to extend Trepan so that it can learn as regression trees and classification trees. A regression tree is a tree-structured model like a decision tree, except that its leaves are characterized by real-valued functions as opposed to predicted classes. The CART algorithm for example, learns regression trees in which a linear function is associated with each leaf. It would be desirable in future work to revamp the algorithm to learn as regression trees.

Inventory management is a field which is shaped by factors as forecasting accuracy of demand and lead time. If the factors and relationship among variables are determined and the scenario is understood better, the inventory management can be enhanced and the entire logistics flow can be improved.

Author's Publications

C. A. Mitrea and C. K. M. Lee “Design and Development of a Knowledge Discovery System in Inventory Management”, *The IEEE International Conference on Industrial Engineering and Engineering Management*, Hong-Kong, 2009

C. A. Mitrea, C. K. M. Lee and Zhang Wu, “A Comparison between Neural Networks and Traditional Forecasting Methods: A Case Study”, *International Journal of Electronic Business Management*, 2009, Vol, 1, No. 2, p 67-72

References

- [1] Zytkow J. and Herbert S. - “A theory of historical discovery: the construction of computational models”, in *Machine Learning*, pp 107-137, 2004
- [2] Michalski, R. S. –“A theory and methodology of inductive learning”, in *Machine Learning: An Artificial Intelligence Approach*, pp 82-134, 1983
- [3] Piatetski-Shapiro, W. Frawley – “Knowledge discovery in databases”, in *Knowledge Discovery in Databases 1991*, pp. 1--30
- [4] Fayyad, U., G. Piatetski-Shapiro, P. Smyth, and R. Uthurusamy. –“Advances in Knowledge Discovery and Data-mining”, *AAAI/MIT Press*, 1996
- [5] Andrews, R., Diederich, J., & Tickle, A.B., - “Survey and critique of techniques for extracting rules from trained artificial neural networks”, in *Knowledge-Based Systems* 8 (6), 373-389, 1995.
- [6] Towell, C.G and F. Gomide. - “Extracting refined rules from knowledge –based neural networks” *Machine Learning* vol. 13, no. 1, 1993
- [7] Andrews, R, J. Dieterich - “Survey and critique of techniques for extracting rules from trained artificial neural networks “in *Knowledge-Based Systems*, 1996
- [8] Fu, L.M - “Rule generation from neural networks” *IEEE Transactions Systems*, 1994
- [9] Lambert, D.M and Stock J.R .-” *Strategic Logistics Management*”, 3rd edition, Irwin, 1993
- [10] Ronald H. Ballou –“Business Logistics Management” Prentice-Hall: Upper Saddle River, NJ, 1999. (4th Edition)
- [11] Liberman B.M, Susan Helper –“The empirical determinants of inventory levels in high-volume manufacturing”, *International Journal of production economic*, 1998
- [12] Vries J. –“The complex relationship between inventory control and organizational settings: theory and practice” *International Journal of Production Economics*, Vol. 93-94 pp.273-84, 2005
- [13] Rajev N –“Inventory management performance in Indian machine tool SMEs: what factors do influence them?” *IEEM*, 2008
- [14] Huson, M., Nanda, D., –“The impact of just-in-time manufacturing on firm performance in the US”, *Journal of Operations Management* 12 (3, 4), 297–311, 1995.

References

- [15] Watts, C.A., Hahn, C.K., Sohn, B., -“Monitoring the performance of a reorder point system: A control chart approach” *International Journal of Operations and Production Management* 14 (2), 51–62, 1994.
- [16] Yi-Feng Hung, Hsinchu -“Determining Safety Stock for Semiconductor Manufacturing [D]” National Tsing Hua University, 2001
- [17] Yu Jinping, Weng Falu. “The Research about the safety stock Based on the BP neural network ” *Agriculture Network Information* 2004.11:24-26
- [18] Wang Dongxu, Shen Yimin, Wang Zhiqiang. “A Model of Forecasting Safety Stock in ERP Based on BP Neural Network. ” *Journal of Computer Applications*. 2001: 53-55
- [19] He Jia, Wang Ziniu, Luo Gang, Zhang Nan. “Research about the Forecasting of Safety Stock Based on Chaotic Neural Network”, 2007
- [20] Schary P.B. -“Logistics Decisions-Text and Cases”, The Dryden Press, USA, 1984
- [21] Bowersox D.J, Closs, D.J, and Helferich O.K -“Logistical Management”, MacMillan, New York, 1986
- [22] Magee JF, Copacino WC, and Rosenfield DB -“Modern Logistics Management”, Wiley , New York, 1985
- [23] Makridakis S and Wheelwright SC -“Forecasting issues and challenges for marketing management “, *Journal of Marketing*, 55:24-37, 1977
- [24] Boldt BI, Jr-“Sound business forecasting”, Today’s Executive, 5(1):6-11, 1982
- [25] S. Chopra & P. Meindl- “Supply Chain Management- Strategy, Planning & Operations” 3rd edition, pp330 (safety stock case), 2007
- [26] Wang, George C. Jain, Chaman L. -"Regression Analysis Modeling & Forecasting", 2003
- [27] G.E.P. Box and F.M. Jenkins. -“Time Series Analysis: Forecasting and Control. Holder-Day, San Francisco, 1976
- [28] C. Chatfield.-” The analysis of time series” Chapman & Hall, London 1989
- [29] N.R Draper and H. Smith -“Applied Regression Analysis” John Wiley & Sons, New York, 1981
- [30] E. Enders. -“Applied Econometric Time Series”, John Wiley & Sons, New York, 1994
- [31] J. Hamilton -“Time Series Analysis”, Princeton University Press, Princeton, NJ, 1994

References

- [32] R.B Litterman. “Forecasting with Bayesian Vector Autoregressions-five years of experience” *Journal of Business and Economics*, 4(1): 25-37, 1986
- [33] <http://www.duke.edu/~rnau/411arim.htm>
- [34] M.B Priestley. “Non-Linear and Non-Stationary Time Series Analysis”, Academic Press, London, 1988
- [35] H. Tong “Non-Linear Time Series-A Dynamic System Approach “ ,Oxford University Press, Oxford, UK,1990
- [36] Saaty TL –“Priority setting in complex problems”, *IEEE Transactions in Engineering Management*, 144-150, 1983
- [37] Dyer RF and Forman EH –“Analytic approach to Marketing Decisions”, Prentice-Hall, Englewood Cliffs NJ, 1991
- [38] Saaty TL and Vargas LG –“Prediction, Projection and Forecasting”, Kluwer Academic Publishers, Dordrecht, 1991
- [39] Wolfe C –“How to adjust forecasts with the analytic hierarchy process”, *Journal of Business Forecasting*, 7:13-17, 1988
- [40] Erkki Hakkinen- “Design, Implement and Evaluation of Neural Data Analysis Environment” University of Jyvaskyla, 2001
- [41] Glaser, B and Strauss, A – “The discovery of grounded theory “New York, US, 1967
- [42] Fayyad, U.M., Piatetsky-Shapiro, G., and Smyth, P. - “From Data Mining To Knowledge Discovery”, 1996
- [43] Daniel T. Larose – “Discovering Knowledge in Data: An Introduction to Data Mining “, John Wiley and Son, 2004
- [44] Kohonen, T. –“Self organizing formation of topologically correct feature maps” 1982
- [45] Pawlak, Z, Wong, S.K., Ziarko, W–“Rough sets: Probabilistic versus deterministic approach” *International Journal of Man-Machine Studies archive*. Volume 29, Issue 1, 1988
- [46] Hunt, E. B.-“Concept learning: An information processing Problem” New York, Wiley
- [47] Breiman, L., J.H. Friedman, R. A Olshen –“Classification and Regression Trees”, Belmont, CA: Wadsworth International Group.
- [48] H. Bacha. W. Meyer –“Neural Network architecture for Load Forecasting”, *Neural Networks, IJCNN*, 1992

References

- [49] R.s Gutierrez, A.O Solis and S. Mukhopadhyay—"Lumpy demand forecasting using neural networks", *International Journal of Production Economics*, 2008.
- [50] D.F Specht- "A general regression neural network", *IEEE trans Neural Network*, 1991
- [51] M.R. Amin-Naseri, B. Rostami Tabar – " Neural Networks Approach to Lumpy demand forecasting for spare parts in process industries", *Computer and Communication Engineering. ICCCE*, 2008
- [52] L. Aburto, R. Weber –"Improving supply chain management based on hybrid demand forecasts", *Applied soft. Computing*, pp 136-144, 2007
- [53] Jose M.P. Menezes Jr, Guilherme A. Barreto -"A new look at nonlinear time series prediction with NARX recurrent neural network" *IEEE*, 2006
- [54] Tsungnan Lin, Bill G. Horne, Peter Tino and C. Lee Giles –"Learning long term dependencies in NARX recurrent neural networks", *IEEE Neural Networks*, Volume: 7, Issue: 6, 1996
- [55] Hava T. Siegelmann, Bill G. Horne and C. Lee Giles-"Computational capabilities of recurrent NARX neural network" *IEEE* 1997
- [56] Jose Maria P. Menezes Jr., Guilherme A. Barreto –"Long-time series prediction with NARX network: An empirical evaluation" *Elsevier: Neurocomputing*, 2008
- [57] J. Moody and J. Utans.-"Architecture selection strategies for neural networks: Application to corporate bond rate and prediction", *Neural Networks in the Capital Markets, Refenes A.N. (ed.)*, John Wiley Sons, 1994
- [58] Y. Liao and J. Moody.-"Constructing heterogeneous committees using input feature grouping: application to economic forecasting", *Advances in Neural Information Processing*, 2000
- [59] Michael Negnevitsky – "Artificial intelligence- A guide to intelligence systems", 2nd Edition, 2004
- [60] Andrews, R. J. D, A.B. Tickle. -"A survey and critique of techniques for extracting rules from trained neural networks", *Knowledge-based neural networks*, 1995
- [61] Fu, L.M - "Rule learning by searching on adapted nets", *Proc. of the Ninth National Conference on Artificial Intelligence*, pp. 590-595, 1991
- [62] Towell, G and Shavlik. -"The extraction of refined rules from knowledge based neural networks", *Machine Learning*, 13, pp. 71-101, 1993

References

- [63] McMillan, C., M. C. Morez, and P. Smolensky. -“The connectionist scientist game: rule extraction and refinement in a neural network”, *Proceedings of the 13th Annual Conference of the Cognitive Science Society*, 1991
- [64] Thrun, S.B. -“Extracting provably correct rules from artificial neural network”, *Technical Report IAI-TR-93-5, Institut fur Informatik III Universitat Bonn*, 1994
- [65] Pop, E., R. Hayvard, and J. Diederich -“RULENEG: Extracting Rules from trained ANN by Stepwise Negation”, *QUT NRC technical report*, 1994
- [66] Towell G, Shavlik JW -“The extraction of refined rules from knowledge based neural networks”, *Machine learning* 31:71-101, 1993
- [67] Craven MW, Shavlik JW -“Understanding time series networks.” *International Journal of Neural Systems* 8:373-384, 1997
- [68] M. W. Craven and J. W. Shavlik-“Extracting tree-structured representations of trained networks”, *Advances in Neural Information Processing Systems*, 8, pp 24-30, 1996.
- [69] Rumelhart, D.E., G.E. Hinton, and R.J. Williams, “Learning internal representations by error propagation,” in D.E. Rumelhart and J.L. McClelland, Eds., *Parallel Data Processing, Vol. 1*, Cambridge, MA: The M.I.T. Press, 1986, pp. 318–362.
- [70] Hagan, M.T., and M. Menhaj, “Training feed-forward networks with the Marquardt algorithm,” *IEEE Transactions on Neural Networks*, Vol. 5, No. 6, 1999, pp. 989–993, 1994.
- [71] Moller, M.F., “A scaled conjugate gradient algorithm for fast supervised learning,” *Neural Networks*, Vol. 6, 1993, pp. 525–533.
- [72] Riedmiller, M., and H. Braun, - “A direct adaptive method for faster backpropagation learning: The RPROP algorithm” *Proceedings of the IEEE International Conference on Neural Networks*, 1993.
- [73] Mark W. Craven -"Extracting comprehensible models from trained neural networks", PhD thesis, 1996
- [74] Snyder R. -“Forecasting sales of slow and fast moving inventories”, *European Journal of Operational Research* 140, pp. 684-699, 2002
- [75] Last M., Klein Y., Kandel A. -“Knowledge Discovery in Time Series Databases”, *IEEE Transactions on systems, man, and cybernetics* - part B: Cybernetics, vol. 31, no. 1, 2001

References

- [76] Neale J.J, Tomlin B. T, Willems S. P. –“The role of inventory in superior supply chain performance “, *International Series in Operations Research & Management Science*, 2004

Appendix

APPENDIX

Appendix 1

Sample used for training NARX network in forecasting for time series case.

	64	59	65	73	74	86	68
INPUTS	59	65	73	74	86	68	40
	65	73	74	86	68	40	35
TARGETS	73	74	86	68	40	35	66
Set	1	2	3	4	5	6	7

 ARIMA (1,1,1) output for Time Series forecasting case

Estimates at each iteration

Iteration	SSE		Parameters		
0	16341.0	0.100	0.100	-0.572	
1	13137.9	-0.050	0.250	-0.903	
2	12557.3	0.040	0.400	-0.805	
3	11887.7	0.134	0.550	-0.716	
4	11102.7	0.223	0.700	-0.650	
5	10206.2	0.297	0.850	-0.632	
6	9393.3	0.265	0.925	-0.707	
7	9035.5	0.115	0.979	-0.965	
8	8510.1	-0.012	0.967	-1.237	
9	8494.9	-0.029	0.958	-1.221	
10	8489.7	-0.024	0.963	-1.179	
11	8488.5	-0.025	0.960	-1.187	

Unable to reduce sum of squares any further

Final Estimates of Parameters

Type		Coef	SE Coef	T	P
AR	1	-0.0251	0.2023	-0.12	0.902
MA	1	0.9602	0.1696	5.66	0.000
Constant		-1.1869	0.3311	-3.58	0.001

Differencing: 1 regular difference

Number of observations: Original series 35, after differencing 34

Residuals: SS = 8465.98 (backforecasts excluded)
 MS = 273.10 DF = 31

Modified Box-Pierce (Ljung-Box) Chi-Square statistic

Lag	12	24	36	48
Chi-Square	30.9	37.9	*	*
DF	9	21	*	*

Appendix

P-Value 0.000 0.013 * *

Forecasts from period 35

		95 Percent Limits		Actual
Period	Forecast	Lower	Upper	
36	30.1130	-2.2838	62.5097	

Appendix

Appendix 2

CODE TO TRAIN TS_NET (v 1.3)

```
% INPUTS AND TARGETS DEFINITION
```

```
l1=[64 59 65 73 74 86 68 40 35 66 97 64 75 54 25 70 48 68  
64 35 35 26 51 51 27 48 25 60 26 41 32 37 57 23 ];
```

```
alpha1=[2 1 1 1 1 2 2 2 1 1 2 1 2 2 2 1 2 1  
2 2 1 2 1 1 2 1 2 1 2 1 2 1 1 2 1]';
```

```
l2=[59 65 73 74 86 68 40 35 66 97 64 75 54 25 70 48 68  
64 35 35 26 51 51 27 48 25 60 26 41 32 37 57 23 39];
```

```
alpha2=[1 1 1 1 2 2 2 1 1 2 1 2 2 2 1 2 1 2  
2 1 2 1 1 2 1 2 1 2 1 2 1 2 1 2 2]';
```

```
l3=[65 73 74 86 68 40 35 66 97 64 75 54 25 70 48 68 64  
35 35 26 51 51 27 48 25 60 26 41 32 37 57 23 39 21];
```

```
alpha3=[1 1 1 1 2 2 2 1 1 2 2 2 2 2 1 1 2 1  
2 2 2 1 1 2 2 2 1 1 2 1 2 1 2 2 2]';
```

```
tar_ts=[1 1 1 2 2 2 1 1 2 1 2 2 2 1 2 1 2 2  
1 2 1 1 2 1 2 1 2 1 2 1 1 2 1 2 1];
```

```
% CODING 1 of N (eye -identity matrix)
```

```
% 1 of 2 coding pattern
```

```
I2=eye(2);
```

```
% alpha1 coding
```

```
for i=1:34
```

```
for j=1:2
```

```
if alpha1(i,1)==1
```

```
cod_alpha1(i,j)=I2(1,j);
```

```
end;
```

```
if alpha1(i,1)==2
```

```
cod_alpha1(i,j)=I2(2,j);
```

```
end;
```

```
end;
```

```
end;
```

```
%alpha2 coding
```

```
for i=1:34
```

```
for j=1:2
```

```
if alpha2(i,1)==1
```

```
cod_alpha2(i,j)=I2(1,j);
```

```
end;
```

```
if alpha2(i,1)==2
```

```
cod_alpha2(i,j)=I2(2,j);
```

```
end;
```

```
end;
```

```
end;
```

```
% alpha 3 coding
```

```
for i=1:34
```

```
for j=1:2
```

```
if alpha3(i,1)==1
```

```
cod_alpha3(i,j)=I2(1,j);
```

```
end;
```

```
if alpha3(i,1)==2
```


Appendix

```

cod_alpha3(i,j)=I2(2,j);
end;
end;
end;
% calculate transposes
cod_alpha1=cod_alpha1';
cod_alpha2=cod_alpha2';
cod_alpha3=cod_alpha3';
% CREATE INPUT MATRIX
in_ts=[l1;cod_alpha1;l2;cod_alpha2;l3;cod_alpha3];
% CREATE TOPOLOGY AND ARCHITECTURE OF NN
ts_net=newff(in_ts,tar_ts,20,{'tansig','purelin'},'trainrp','learngdm','mse',{'fixunknowns','removeconstantrows'},{'removeconstantrows'},'dividerand');
% TRAINING PARAMETERS
inv_net.trainParam.epochs=1000;
inv_net.trainParam.show=10;
inv_net.trainParam.goal=0;
inv_net.trainParam.min_grad=1e-010;
inv_net.trainParam.max_fail=6; % error of the validation set (no of iterations)
inv_net.trainParam.delt_inc=1.2;
inv_net.trainParam.delt_dec=0.5;
inv_net.trainParam.delta0=0.07;
inv_net.trainParam.deltamax=60;
% TRAIN NN
ts_net=train(ts_net,in_ts,tar_ts);
% EXTRACT WEIGHTS AND BIASES
w1_ts_net=ts_net.iw{1,1};
w2_ts_net=ts_net.lw{2,1};
b1_ts_net=ts_net.b{1};
b2_ts_net=ts_net.b{2};
% CREATE MATRIX FOR TREPAN
in_trepan=[l1;alpha1';l2;alpha2';l3;alpha3';tar_ts]';
% Y=INV_NET(X)
function y = ts_net_global(x)
global w1_inv_net
global w2_inv_net
global b1_inv_net
global b2_inv_net
a=w1_inv_net*x';
[q,r]=size(a);
% add biases for first layer
for i=1:q
    v1(i,:)=a(i,:)+b1_inv_net(i,:);
end;
% tansig activation in hidden layers
act_v1=tanh(v1);
% add biases for second layer
b=w2_inv_net*act_v1;
act_v2=b+b2_inv_net;
y=((act_v2>0)+1)'; % convert to class down (2) or up (1)

```

The oracle coding in time series case

Appendix

```

% Y=INV_NET(X)
function y = ts_net_global(x)
global w1_inv_net
global w2_inv_net
global b1_inv_net
global b2_inv_net
a=w1_inv_net*x';
[q,r]=size(a);
% add biases for first layer
for i=1:q
    v1(i,:)=a(i,:)+b1_inv_net(i,:);
end;
% tansig activation in hidden layers
act_v1=tanh(v1);
% add biases for second layer
b=w2_inv_net*act_v1;
act_v2=b+b2_inv_net;
y=((act_v2>0)+1)';

```

Sample of training data used by TREPAN in time series case.

instance	l1	alpha1	l2	alpha2	l3	alpha3	class
p-0	64	2	59	1	65	1	up
p-1	59	1	65	1	73	1	up
p-2	65	1	73	1	74	1	up
p-3	73	1	74	1	86	1	down
p-4	74	1	86	2	68	2	down
p-5	86	2	68	2	40	2	down
p-6	68	2	40	2	35	2	up
p-7	40	2	35	1	66	1	up
p-8	35	1	66	1	97	1	down
p-9	66	1	97	2	64	2	up
p-10	97	2	64	1	75	2	down

Appendix

Appendix 3

Tree extracted by TREPAN in time series case.

```
*** ATTRIBUTES ***
```

```
1  l1          I
2  alpha1      N    1    2
3  l2          I
4  alpha2      N    1    2
5  l3          I
6  appha3      N    1    2
7  class       N    up   down
```

```
*** OPTIONS ***
```

```
verbose          : 2
random_seed      : 214002
min_sample_size  : 1000
beam_width       : 5
min_objects      : 20
mofn_alpha       : 0.05
dist_alpha       : 0.1
max_tree_size    : 10
max_splits       : 100
max_mofn_n       : 3
```

```
*** TREE ***
```

```
node      : 1
parent    : 0
child1    : 2
child2    : 3
leaf      : no
pruned    : no
#class 1  : 490(14)
#class 2  : 510(20)
class     : up
priority  : 0.490000
gain      : 1.908425e-001
m         : 2
split     : alpha3=1
split     : 13 > 56
```

```
node      : 2
parent    : 1
child1    : 4
child2    : 5
leaf      : no
pruned    : no
#class 1  : 750(10)
#class 2  : 250(3)
class     : down
```

Appendix

```
priority : 0.145750
gain     : 2.311672e-001
m        : 2
split    : appha3 = 2
split    : alpha1 = 2
```

```
node     : 3
parent   : 1
child1   : 0
child2   : 0
leaf     : yes
pruned   : no
#class 1 : 182(4)
#class 2 : 818(3)
class    : up
priority : 0.075894
```

```
node     : 4
parent   : 2
child1   : 0
child2   : 0
leaf     : yes
pruned   : no
#class 1 : 269(2)
#class 2 : 731(3)
class    : down
priority : 0.028543
```

```
node     : 5
parent   : 2
child1   : 0
child2   : 0
leaf     : yes
pruned   : no
#class 1 : 269(1)
#class 2 : 731(3)
class    : up
priority : 0.008654
```

Appendix 4

The following is the coding in Matlab for training NN used in safety stock case

Appendix

```

% ver 1.5
global w1_inv_net;
global w2_inv_net;
global b1_inv_net;
global b2_inv_net;

% SS determination
S=250; % number of random generated values
demand = unifrnd (1.5,9.5,1,S)*1000;
lead_time=random('unid',12,1,S)+2;
cycle_service_level=unifrnd(0.5000, 0.9999 , 1, S);

for i=1:S
    mean_demand(i)=mean(demand(1:i));
    std_dev_demand(i)=std(demand(1:i));
    mean_lead_time(i)=mean(lead_time(1:i));
    std_dev_lead_time(i)=std(lead_time(1:i));

    std_dev_demand_during_lead_time(i)=sqrt(mean_lead_time(i)*std_dev_demand(
    1,i)*std_dev_demand(1,i)+mean_demand(1,i)*mean_demand(1,i)*std_dev_lead_t
    ime(1,i)*std_dev_lead_time(1,i));
    norm(i)=norminv(cycle_service_level(i));
    safety_stock(i)=norm(i)*std_dev_demand_during_lead_time(i);
    if i==1
        continue;
    else if safety_stock(i)>safety_stock(i-1) % assign class
        class(i)=1; % increase
    else class(i)=-1; %decrease
    end;
end;
end;

% CREATE INPUT MATRIX std_dev_demand(2:S) std_dev_lead_time(2:S)
in_ss_net=[mean_demand(2:S);std_dev_demand(2:S);mean_lead_time(2:S);std_d
ev_lead_time(2:S);safety_stock(1:(S-1));cycle_service_level(2:S)];

% CREATE TOPOLOGY AND ARCHITECTURE OF NN
ss_net=newff(in_ss_net,class(2:S),20,{'tansig','purelin'},'trainscg','lea
rngdm','mse',{'fixunknowns','removeconstantrows'},{'removeconstantrows'},
'dividerand');

% TRAINING PARAMETERS
ss_net.trainParam.epochs=1000;
ss_net.trainParam.show=15;
ss_net.trainParam.goal=0;
ss_net.trainParam.time=Inf;
ss_net.trainParam.min_grad=1e-08;
ss_net.trainParam.max_fail=6; % error of the validation set (no of
iterations)
ss_net.trainParam.sigma=5e-005;
ss_net.trainParam.lambda=5e-006;

% TRAIN NN
ss_net=train(ss_net,in_ss_net,class(2:S));

```

Appendix

```
% EXTRACT WEIGHTS AND BIASES
w1_ss_net=ss_net.iw{1,1};
w2_ss_net=ss_net.lw{2,1};
b1_ss_net=ss_net.b{1};
b2_ss_net=ss_net.b{2};

% CREATE MATRIX FOR TREPAN
in_trepan=[mean_demand(2:S);std_dev_demand(2:S);mean_lead_time(2:S);std_d
ev_lead_time(2:S);safety_stock(1:(S-
1));cycle_service_level(2:S);class(2:S)];
```

-----Coding the oracle

```
% Y=INV_NET(X) - the oracle function used in causal forecasting case
function y = inv_net(x)
global w1_inv_net;
global w2_inv_net;
global b1_inv_net;
global b2_inv_net;
a=w1_inv_net*x';
[q,r]=size(a);
% add biases for first layer
for i=1:q
    v1(i,:)=a(i,:)+b1_inv_net(i,:);
end;
% tansig activation in hidden layers
act_v1=tanh(v1);
% add biases for second layer
b=w2_inv_net*act_v1;
act_v2=b+b2_inv_net;
y=((act_v2>0)+1)'; % output must be 2 for case "increase" and 1 for
class "increase"!!!
```

-----Data sample for safety stock case

p-0	3102.03	1334.07	10	2.8284	0	0.6579	increase
p-1	4988.69	3401.22	7.66667	4.5092	3959.21	0.6528	increase
p-2	4654.73	2856.28	9.25	4.8563	9578.23	0.5872	decrease
p-3	5594	3244.97	8.2	4.8166	5333.12	0.5831	increase
p-4	5197.14	3060.85	8.16667	4.3089	5980.67	0.5288	decrease
p-5	5234.94	2795.95	7.42857	4.3916	1737.3	0.9682	increase
p-6	4945.98	2714.51	6.875	4.3569	44913.5	0.6694	decrease
p-7	4897.34	2543.38	7.66667	4.717	9947.25	0.8464	increase
p-8	5131.2	2509.37	7.8	4.4672	24659.5	0.9664	increase
p-9	5226.27	2401.39	7.45455	4.3901	43874.4	0.9463	decrease
p-10	5384.93	2354.67	7.66667	4.2498	38409.9	0.6771	decrease

Appendix

Appendix 5

Tree obtain using TREPAN for safety stock case

```
*** ATTRIBUTES ***
```

```
1    mean_demand R
2    std_dev_demand R
3    mean_lead_time R
4    std_dev_lead_time R
5    previous_safety_stock R
6    cycle_service_level R
7    class N increase decrease
```

```
*** OPTIONS ***
```

```
verbose          : 2
random_seed      : 214338
min_sample_size  : 1000
beam_width       : 5
min_objects      : 20
mofn_alpha       : 0.05
dist_alpha       : 0.1
max_tree_size    : 8
max_splits       : 100
max_mofn_n       : 3
```

```
*** TREE ***
```

```
node      : 1
parent    : 0
child1    : 2
child2    : 3
leaf      : no
pruned    : no
#class 1  : 557(132)
#class 2  : 443(117)
class     : increase
priority  : 0.443000
gain      : 2.227284e-001
m         : 1
split     : cycle_service_level < 0.705302
```

```
node      : 2
parent    : 1
child1    : 8
child2    : 9
leaf      : yes
pruned    : no
#class 1  : 850(84)
#class 2  : 150(19)
class     : increase
priority  : 0.062850
```

Appendix

```
node      : 3
parent    : 1
child1    : 4
child2    : 5
leaf      : no
pruned    : no
#class 1  : 314(48)
#class 2  : 686(98)
class     : decrease
priority  : 0.182434
gain      : 2.676270e-001
m         : 1
split     : previous_safety_stock < 8.544524
```

```
node      : 4
parent    : 3
child1    : 6
child2    : 7
leaf      : no
pruned    : no
#class 1  : 158(7)
#class 2  : 842(94)
class     : decrease
priority  : 0.066737
gain      : 1.200468e-001
m         : 1
split     : mean_demand < 2.086436
```

```
node      : 5
parent    : 3
child1    : 0
child2    : 0
leaf      : yes
pruned    : no
#class 1  : 748(41)
#class 2  : 252(4)
class     : increase
priority  : 0.039970
```

```
node      : 6
parent    : 4
child1    : 0
child2    : 0
leaf      : yes
pruned    : no
#class 1  : 627(0)
#class 2  : 373(5)
class     : increase
priority  : 0.011029
```

```
node      : 7
parent    : 4
child1    : 0
child2    : 0
leaf      : yes
pruned    : no
#class 1  : 109(7)
```


Appendix

```
#class 2 : 891(89)
class    : decrease
priority : 0.042817
```

```
node     : 8
parent   : 2
child1    : 0
child2    : 0
leaf      : yes
pruned    : yes
#class 1 : 647(16)
#class 2 : 353(17)
class     : increase
priority  : 0.040970
```

```
node     : 9
parent   : 2
child1    : 0
child2    : 0
leaf      : yes
pruned    : yes
#class 1 : 916(68)
#class 2 : 84(2)
class     : increase
priority  : 0.025447
```

Appendix

Appendix 6

Sample of inventory database from Panasonic Refrigeration Devices
Company located in Singapore.

Item	Mth end Inv.	Sub-con. Inventory	May			Jun			Jul	
			Order	Prodn	Inv.	Order	Prodn	Inv.	Order	Prodn
SQ-0230991 (275)		C 31138	155365	18690 - 2/5		104685			113819	
Hot Rolled Coil		B 84240	B 40000	(46643)					B 40000	(47914)
2.3 x 991 SPHD		P 6890	B 18120	(46102) - 2/5						
1.247kg/pc			C 74575	(46644/46106) - 2/5						
(1.8 mth stock)				28532	226431		39481	186950		45703
2/5	99070		C 92460	100000	5.74	C 100000	100000	4.09	C 100000	100000
U/SHELL	(46645C-P080122M)		23/5-(47227)	(P080225M-PRDM)		(47918)	(P080319M-PRDM)		(48590)	(P08042)
SQ-0231013 (281)		A	-20478			-92201			-56210	
Hot Rolled Coil		P	B 61950	(46643) - 2/5		C 110000	(47917)		B 100000	(47914)
2.3 x 1013 SPHD		B 9480	C 241075	(47226) - 23/5		P 120000	(47919)		C 90000	(48587)
1.302kg/pc		C 239120	C 33975	(46106) - 2/5						
(1.8 mth stock)		J		228593	357007		209714	377293		260830
U/SHELL			A 96810	100000	1.70	A 50000	50000	1.45	A 100000	100000
			10/5-(47231)	(P080225M-PRDM)		(47915)	(P080319M-PRDM)		(48622)	(P08042)
SQ-0231049 (291)		B	140999			33555			55468	
Hot Rolled Coil		C 71070	B 38150	(46643) - 2/5					C 40000	(48587)
2.3 x 1049 SPHD		P 7820	C 58714	(46644) - 2/5					P 110000	(48625)
1.398kg/pc			127167	(return from PRDM)						from 3.4
(2.2 mth stock)		J		91122	211799		32182	179617		66392
U/SHELL	150000			300000	6.58	400000	400000	2.71	350000	350000
	(46658P-P080122M)			(P080225M-PRDM)		(47929)	(P080319M-PRDM)			(P08042)
SQ-0281049 (291)		P 7690	22167.6			9610.2			9198	
Hot Rolled Coil		A 14700	P 20000	(46657)						
2.8 x 1049 SPHD										
1.639kg/pc										
(2.2 mth stock)				10703	31687		4327	27360		8064
U/SHELL					7.32	J 20000	20000	3.39		
						(47929)	(P080319M-PRDM)			
SQ-0360991(275)		P 9950	27164			31030			16956	
Hot Rolled Coil		J 32900	J 43870	(47232) - 22/5		J 50000	(47922)			
3.6 x 275 SPHD			P 120000	(46115)						
1.678kg/pc										
(1.8 mth stock)				66835	139885		62623	127262		53462
U/SHELL					2.23	U		2.38		
SQ-0270991 (275)		C 81415	-34103			7500			25	
Hot Rolled Coil		J				C 70000	(47917)		C 20000	(48587)
2.7 x 991 SPHD		P								
1.538kg/pc		B								
(1.8 mth stock)				30914	50501		47002	73499		36666
U/SHELL (80%)					1.07	P		2.00		
L/SHELL (20%)						J				