USING MACHINE LEARNING TECHNIQUES FOR EARLY COST ESTIMATION OF STRUCTURAL SYSTEMS OF BUILDINGS

A Thesis Submitted to
the Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in Architecture

by
Sevgi Zeynep DOĞAN

September 2005
İZMİR

We approve the thesis of **Sevgi Zeynep DOĞAN**

Date of Signature

.....
Assoc. Prof. Dr. H. Murat GÜNAYDIN
Supervisor
Department of Architecture
İzmir Institute of Technology

28 September 2005

.....
Prof. Dr. Gökmen TAYFUR
Co-Supervisor
Department of Civil Engineering
İzmir Institute of Technology

28 September 2005

.....
Prof. Dr. David ARDITI
Department of Civil and Architectural Engineering
Illinois Institute of Technology

28 September 2005

.....
Assist. Prof. Dr. Emre ERGÜL
Department of Architecture
İzmir Institute of Technology

28 September 2005

.....
Assist. Prof. Dr. Yavuz DUVARCI
Department of City and Regional Planning
İzmir Institute of Technology

28 September 2005

.....
Assoc. Prof. Dr. H. Murat GÜNAYDIN
Head of Department
İzmir Institute of Technology

28 September 2005

.....
Assoc. Prof. Dr. Semahat ÖZDEMİR
Head of the Graduate School

ACKNOWLEDGMENTS

I would like to gratefully acknowledge the enthusiastic and patient supervision of my advisor Dr. H. Murat Günaydın. Dr. Günaydın was not only my academic advisor but also my mentor throughout the research study. My deepest gratitude also goes to Prof. Dr. David Arditi of IIT in Chicago who has also been a supervisor and mentor to me. I could not have imagined having better advisors and mentors for my PhD; without their wisdom, knowledge and insightfulness this dissertation could not have been written. Their dedication to reviewing my work and helping to improve it were invaluable. I would also like to thank Prof. Dr. Gökmen Tayfur for his excellent course on artificial neural networks and his careful reviewing and feedback. I gratefully acknowledge the contribution of my thesis committee composed of Dr. Yavuz Duvarcı and Dr. Emre Ergül. I also want to thank İzmir Institute of Technology for the financial support it provided during the progression of this thesis.

Last but not least, I am forever indebted to my mom Firuz and my dad Kiper for their understanding, endless patience, encouragement, and financial and moral support when they were most required.

ABSTRACT

It is desirable to predict construction costs in the early design stages in order to make sure that target costs are met and competitive prices are realized. This study investigates the possibility of predicting the cost of construction early in the design phase by using machine learning (ML) techniques. To achieve this objective, artificial neural network (ANN) and case based reasoning (CBR) prediction models were developed in a spreadsheet-based format. An investigation of the impacts of weight generation methods on the ANN and CBR models was conducted. The performance of the ANN model was enhanced by experimenting with the weight generation methods of simplex optimization, back propagation training, and genetic algorithms while the CBR model was augmented by feature counting, gradient descent, genetic algorithms (GA), decision tree methods of binary-dtree, info-top and info-dtree.

Cost data belonging to the superstructure of low-rise residential buildings were used to test these models. It was found that both approaches were capable of providing high prediction accuracy, 96% for ANN using simplex optimization for weight determination, and 84% for CBR using GA for attribute weight selection. A comparison of the Excel-based ANN and CBR models was made in terms of prediction accuracy, preprocessing effort, explanatory value, improvement potentials and ease of use. The study demonstrated the practicality of using spreadsheets in developing ANN and CBR models for use in construction management as well as the potential benefits of enhancing ANN and CBR models by using different weight generation methods.

ÖZET

Maliyet tahmini, yapım projesinin tasarım sürecine ait erken evre için çok önemlidir. Bu çalışmada, otomatik öğrenme tekniklerinden ikisinin, yapay sinir ağları (YSA) ve vaka tabanlı gerekçeleme (VTG)'nin, bina tasarım sürecinin erken evresinde yapılan maliyet tahmini için uygunluğu ve başarısı araştırılmıştır. Hem YSA hem de VTG'nin elektronik tablo simülasyonları geliştirilmiş ve maliyet tahmin modelleri oluşturulmuştur. İnşaa edilmiş konut projelerine ait maliyet verisi modellerin örnek uygulamasında kullanılmıştır. Çeşitli ağırlık üretim yöntemlerinin YSA ve VTG modellerinin tahmin doğruluğu üzerindeki etkisi konut projelerine ait maliyet tahmini örneğinde araştırılmıştır. YSA için geriye yayılma eğitimine alternatif olarak, genetik algoritmalar ve simpleks optimizasyonu metodu; VTG için ise özellik sayma, genetik algoritmalar ve gradyan iniş metodları ile karar ağaçlarından türetilen üç farklı yöntem kullanılmıştır. YSA modeli ağ ağırlıklarının belirlenmesinde simpleks optimizasyonunu kullandığında %96 başarı oranı; VTG modeli özelliklerin ağırlıklarını genetik algoritmaları kullanarak seçtiğinde %84 başarı oranı yakalamıştır.

YSA ve VTG'nin elektronik tablo şeklinde geliştirilen maliyet tahmin modelleri işlem öncesi çaba, açıklanabilirlik değeri, doğruluk oranı, gelişme potansiyeli ve kullanım kolaylığı açısından karşılaştırılmıştır. Modellerin elektronik tablo simülasyonları şeklinde geliştirilmiş olması modellerdeki ağırlık üretim değişiklilerini yapabilmek için esneklik sağlamış ve modellerinin daha fazla gelişimine olanak vermiştir.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

During construction and design of buildings, cost prediction is necessary to make sure that target costs are met and competitive prices are realized. Conventional cost prediction methods generally require quite detailed information about the building project. It is therefore only in a late phase of the construction process that the available knowledge suffices to base cost-related decisions on it. However, the architects' influence on cost decreases over time because decisions with substantial cost impact are made in the early phases.

The properties of emerging machine learning (ML) methods give rise to the hope that they can support cost prediction in an early phase of design as they are able to detect relationships between conceptual design information and cost unknown to architects (i.e. artificial neural networks) or as they are able to use analogy-based techniques by storing and retrieving previous design solutions (i.e. case-based reasoning). This study investigates the possibilities to support early design phase of building project design with machine learning techniques. Cost prediction as a prototypical architectural design activity and construction management application is chosen for the following reasons:

1.1. Motivation of the Research

Design is the basic business of the architect. However, architects are also required to be knowledgeable in the areas of construction methods, sequencing of construction, and cost. Consideration of these factors is inherently a part of the design. Yet, design is also the major determinant of original and operating building cost. It is possible for architects and construction managers to exert a highly acceptable degree of control over the cost of building design and construction process.

Design and cost cannot be separated if cost control is to be effective. An important fact stated for the architects at the outset is that cost is not likely to be

controlled as well as may be done when this effort is divorced from the design process itself or when, in fact, cost control is not a part of the designer's philosophy (Heery, 1975).

Early project cost estimates are significant to a client because they need to be accurate enough to impart the confidence needed to commit additional funds to the project. However, architects are often charged with not being able to predict with reasonable accuracy the cost of a building project that they design. Thus, an architect's services are engaged only when luxury can be afforded. Reports of building costs exceeding the architect's estimate contribute to the persistence of this image. The implication is that if one employs an architect, one will not know the cost until it is too late to do anything about it. Architects often fail to consider carefully enough the potential costs of a project, or they do not take any responsibility for predicting and controlling costs at the early design stage. However, there is a professional standard and responsibility to be met and cost consideration is among the most basic ones.

Architects should assure that the project is built within the cost forecast (Hunt, 1967). Therefore, they should have the knowledge and techniques to accurately predict the cost of any size and type of building project and keep the cost under control. However, the current cost prediction practice in building projects is inadequate and unreliable. This may be due to incomplete and fuzzy nature of inputs and outputs of design and construction work. Because of the importance of factors that could not be quantified, the current decision support systems have little chance of success. Newly emerging machine learning (ML)-based cost models offer some methods that may provide architects with the results that they are looking for. The use of such models has outperformed traditional highly subjective or highly objective procedures. Objective (mathematical and quantitative analysis) techniques have the ability to identify elements in an explicit manner and subjective (judgment) techniques have the advantage of developing procedures based on experience and intuition incorporating up-to-date knowledge and feelings about the project to obtain current costs. However, a good prediction technique should include both historical data and construction experience and knowledge. The ML-based prediction methods proposed in this study use artificial neural networks (ANN) and case based reasoning (CBR) methods to provide an estimate that includes both objective and subjective information. These artificial-intelligence based techniques either emulate the human ability to learn from past experience and to apply quick solutions to new situations or use analogy-based

decisions to propose new solutions. The proposed techniques use the judgment process of experienced estimators to develop prediction models.

This dissertation is investigating the usefulness of ML-models in assisting and improving the performance of architects and construction managers who are responsible for predicting building costs in their early design process.

The development of ML-cost models and their wider application to early design and construction processes have the following advantages:

1. It is possible to produce suitable cost information at an early stage in the design process.
2. This information is more reliable, introducing greater confidence into the decision-making process.
3. More information is generated so that better informed decisions are made.
4. The cost information is provided more quickly.

Architects and construction managers need to be aware of the merits of ML-based cost models in order to adopt them for early cost prediction.

1.2. Objectives and Organization of the Thesis

This study aims to see if higher cost prediction rates at the early design stage can be obtained by using ML techniques. Thus, the goals of this research are twofold. The first goal is of practical nature and involves empirically predicting the unit cost of a structural system as accurately as possible at the early phase of design by using ML techniques of artificial neural networks (ANN) and case based reasoning (CBR). The second goal is academic and involves improving and comparing the efficacy of these ML models.

In order to reach these goals, first cost prediction and ML techniques are reviewed in Chapter II, then the methodology of the study is described in Chapter III. Models developed by using ANN and CBR techniques and enhanced by various weight generation methods are explained in this chapter. Next, data pertaining to the early design parameters and unit cost of the superstructure of residential building projects are used to test ANN and CBR models. Chapter IV contains test results of the cost data run in ML models. Findings are analyzed and a comparison is made in terms of prediction accuracy, preprocessing effort, explanatory value, improvement potentials and ease of

use. Chapter V presents the conclusions and recommendations for further studies. Figure 1.1 presents the methodology of the thesis.

Hegazy and Ayed's (1998) spreadsheet-based ANN prediction model and a commercial software NeuroSolutions (2002) are used for the ANN modeling. In order to determine ANN weights, as an alternative to back-propagation training of NeuroSolutions, two other techniques named simplex optimization, and genetic algorithms (GA) are used. Simplex optimization and GA are implemented using Excel add-in programs of Solver and Evolver (1998), respectively.

The CBR prediction model is established by developing an Excel-based simulation. The model is assigned attribute weights by six different weight generation methods in order to compare their impact on the performance of CBR prediction. Weights for attributes of the CBR-Excel model are computed by (1) the feature counting method, (2) the gradient descent method, (3) genetic algorithms (GA), (4) binary-dtree method, (5) info-top method, and (6) info-dtree method.

Three commercial software help to determine weights in the CBR-Excel model. A CBR software called Esteem (1996) is used to implement the gradient descent weight generation method. Evolver (1998) is used once more for GA computations. Binary-dtree, info-top and info-dtree methods named by Ling et al. (1997) are adapted by using induction decision trees (ID3). The decision tree of the cost prediction problem is constructed by using the See5 software (1997).

Boosted decision trees (BDT) constructed by See5 is used as the third machine learning technique, as an alternative to ANN and CBR models, for the cost prediction problem at hand. However, cost data had to be classified into a large number of classes because of the BDT modeling rules. In these circumstances, few number of data available in this study produced outcomes that were less accurate than the ANN and CBR outcomes. Therefore, the BDT model and the results achieved are presented in Appendix B.

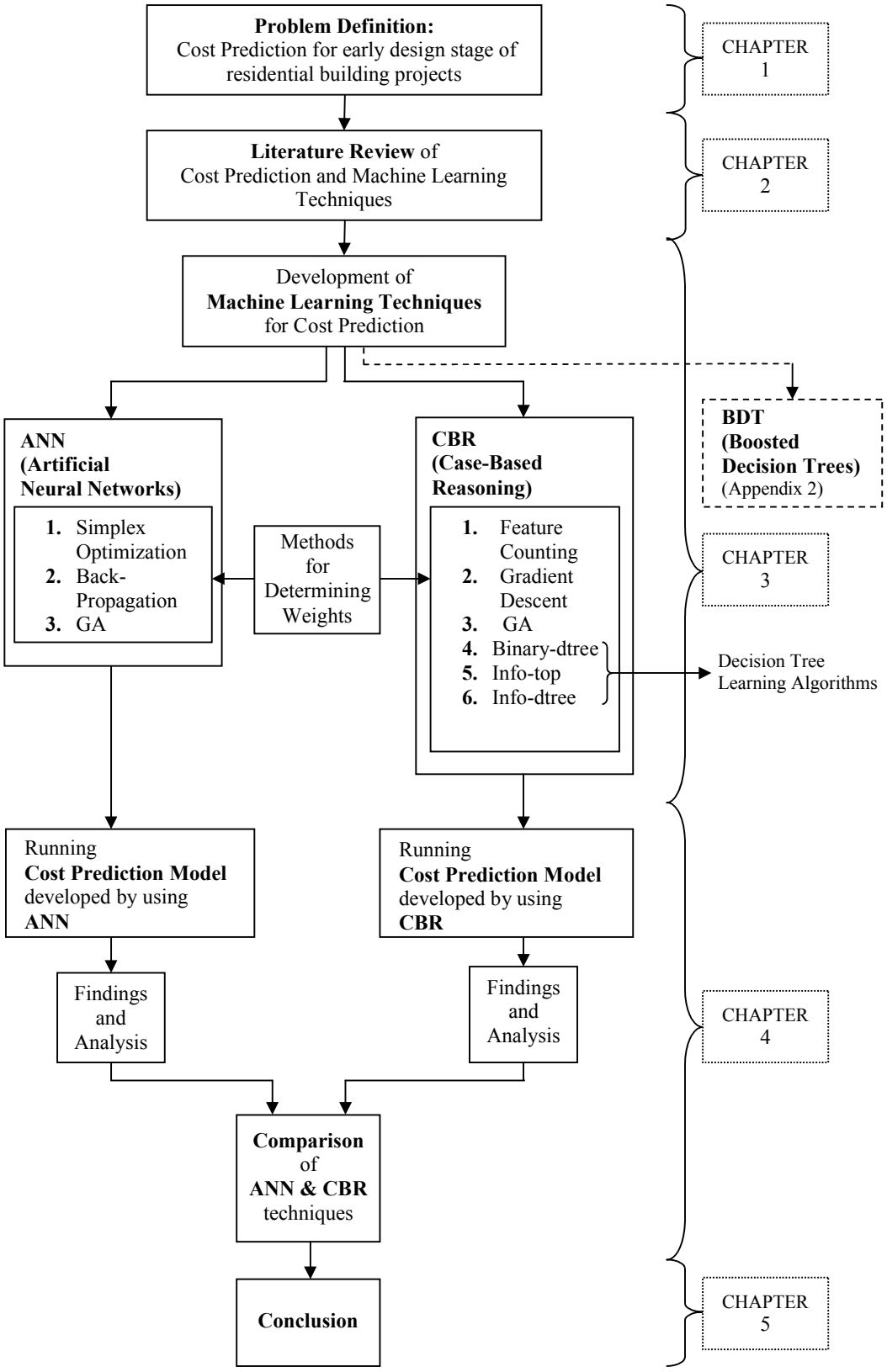


Figure 1.1. Methodology of the study

CHAPTER 2

COST PREDICTION AND MACHINE LEARNING TECHNIQUES

Cost is one of the major criteria in decision making at the early stages of a building design process. In today's globally competitive world, diminishing profit margins and decreasing market shares, cost control plays a major role for being competitive while maintaining high quality levels. To this end, designers and project managers use a number of cost prediction techniques and intuitive judgments by utilizing both their experience and data from previous projects.

Developments in computer and software technology have facilitated novel approaches for cost prediction. By the emergence of computerized learning techniques named machine learning (ML) tools (i.e., artificial neural networks, case-based reasoning, decision trees) more effective predictive models can now be investigated. These techniques have proven to be valuable tools in a wide range of applications. Business decision support and data-mining are few of them. These techniques share one common feature: A solution is learned, then that solution is applied in a manner to make useful predictions (Francone 1999). Prediction involves estimating the unknown value of an attribute of a system under study given the values of other measured attributes (Friedman 2003). In predictive (machine) learning the prediction rule is derived from data consisting of previously solved cases (Friedman 2003). For the construction industry which is highly experience-oriented, construction problems mostly come with previous data of similar cases. Therefore, this study provides insights into integrating two currently separate research areas. Integration of cost prediction in the early design stage of the construction process; and artificial neural networks (ANN) and case based reasoning (CBR) tools of machine learning are investigated.

Therefore, traditional cost prediction techniques, basic ANN and CBR processing and a literature review of ANN and CBR application to cost prediction have been covered. To this end, this chapter is composed of two main sections. The first

section includes a review of cost prediction and its techniques. The second section presents an introduction to machine learning and reviews the basics of ANN and CBR.

2.1. Cost Prediction

The cost prediction function inherent in architectural design is a complex basic component, which can be performed at different subphases of the process (i.e., inception, design, construction, operation and maintenance); however a project's optimization can be best obtained by experimenting with different design variables, one at a time, at the predesign (early design) stage (Siqueira 1999a, 1999b, Seyyar 2000). This is the stage where decisions have the greatest impact on project cost, schedule and performance. Clearly, this stage is the most crucial for meeting a project's criteria, and front-end cost estimating is vital in project development. Estimates prepared at this early stage, accordingly, form the basis for analyses of return on investment and, assist owners and their agents in making go-no-go decisions.

At this stage, cost estimation depends on very little information available, therefore a high level of uncertainty characterizes this phase of the process. This study focuses on such cost estimating models. Front-end, parametric, conceptual, early design and/or order of magnitude cost estimates refer to estimates using the main parameters of a project to predict its cost. These estimates are used to assist in go-no-go decisions while minimizing estimating efforts spent on non-viable projects (Melin 1994, Paek 1994, Barrie and Paulson 1992, Carr 1989, Karshenas 1984).

The generation of the early design estimates to a reasonable degree of accuracy, in a timely fashion, can be considerably complex. The highly unstructured nature of these estimates, as well as the many different estimating practices, may be the reason why inconsistent and therefore unreliable cost estimates are obtained in the process of estimating the cost of building construction. Cost estimators often deal with a wide variety of design parameters. The building construction industry, like manufacturing industries is on the outlook for high quality and efficiency. However the output of the construction activity is large in scope and cost, and unique in the sense that it is producible only for a particular occasion; thus the concept of building a prototype does not apply in the construction industry because it is not practical and economical (Feery

and Brandon 1984). Detailed cost estimating, therefore, is currently used for each and every construction project, making estimation a time consuming and costly process.

While this practice is well suited to market conditions in the past years, it has become inadequate for the current industry's needs (i.e., tough competition, limited resources, decreasing profit margins, etc.). The increase in human population and production activities have led to the emergence of more complex building construction projects in an ever more competitive market. This has made the targets of the time-cost and quality triangle even harder to achieve. For instance, in case of a building project defined by four design parameters, the consideration of three different values for each parameter, varying one parameter at a time, would generate 81 (3^4) different project alternatives, and, as such, may require the generation of 81 detailed cost estimates.

The above example illustrates that designers cannot achieve optimal solutions in a timely and cost effective manner, through the generation of detailed cost estimates for different scenarios (i.e., involving different design parameters) for each Request For Proposal (RFP). The time and cost involved in preparing such estimates are prohibitive for planning purposes. The result is, in most cases, that designers develop project proposals by just considering limited scenarios, therefore probably being far from optimal. Then, the goal of defining a project of minimum cost while meeting defined criteria may not simply be achieved. A solution for this problem is, therefore, to automate the cost estimating process, in such a way to allow for 1) interactive (owner and designer) project scope definition, 2) the timely generation of what-if-type scenarios, 3) reliable cost estimate to assist in go-no-go decisions, and 4) an open and flexible cost estimating environment capable of benefiting from actual costs incurred on previous projects.

Optimizing the cost estimating process means determining the best tools and system to be used for that end. Realizing that, construction companies are looking for new concepts and advanced tools to assist the optimization of the cost estimating process. At this point, the developments in communication and information technologies and applications of computer aided design methods show invaluable benefits and opportunities. As such, automation for cost estimating may bring the efficiency and accuracy so needed in delivering a number of alternative cost estimates generated in a timely and cost effective manner. Increased efficiency and accuracy in cost estimating provide companies with a competitive edge. The integration of

predesign cost estimating principles with machine learning techniques to develop a methodology capable of responding to these needs is investigated in this study.

2.1.1. Cost Prediction Techniques

In an environment of developing technology, global competition, harsh economical factors and inevitable rapid movement necessity, the complexity of construction projects are impossible to overcome by conventional methods and applications. On the other hand, the increasing capacity of information technologies along with the diminishing costs of hardware and software systems make computer-aided methods more appealing than ever. Parallel to this, traditional cost-estimating techniques –known as single price estimating models (unit, volume, area and storey enclosure method), elemental estimating, operational estimating and resource related methods –are also replaced by advanced cost estimating systems –known as casual-empirical models, regression models, simulation models and expert systems –that use hardware and software to convert data into appropriate information for the ultimate users (Orhon et al. 1986, Seyyar 2000). Conventional manual methods lost their effectiveness in terms of ease, accuracy and time management when compared with advanced computer-aided cost estimating applications (Yaylagül 1994). When projects become larger in scope and complexity, cost estimation models take many parameters into consideration utilizing computers for storing, processing and transferring of various data. Today, many construction firms use computer aided cost estimation systems designed for better cost estimation performance in their projects and their organizations (Seyyar 2000). The importance of computers and computer aided cost estimation systems in the fast and easy determination of the interaction between design parameters and cost; in eliminating the complexities in the cost estimation process and providing automation, cannot be overlooked and underestimated (Seyyar 2000). However, these models are still not sufficient and feasible enough for the early architectural design stage. The emergence of machine learning techniques promise further achievement for early design cost prediction.

The accuracy of any prediction depends on the amount of information available at the time of the prediction. As stated in the Construction Industry Institute's "Improving Early Estimates' (CII 1998), "... any cost estimate is assigned a range of

accuracy (±percentage). These ranges narrow as the quantity and quality of information increase through the life of a project. This implies that estimate accuracy is a function of available information, a generally accepted fact in engineering and construction." Good estimating practice and experienced personnel are also found to have considerable impact on estimating accuracy, especially on preliminary estimates, since at this stage available information scarce and often poorly defined (CII, 1998).

CII's (1998) study highlights the following as major factors impacting estimates' accuracy:

1. Quality and amount of information available for preparing the estimate
2. Time allocated to prepare the estimate
3. Proficiency and experience of the estimator and the estimating team
4. Tools, techniques and models used in preparing the estimate

Accordingly, estimates are classified and their corresponding range of accuracy is defined. The cost estimate classifications adopted by the Association for Advancement of Cost Engineering (AACE) International and the Construction Industry Institute (CII), are shown in Table 2.1 and Table 2.2, respectively.

This study will focus on estimates prepared at a predesign stage, when the level of project definition is within 10 to 40%. The expected accuracy range for these estimates is between -20 to +30% in AACE's classification (see Table 2.2)

A preliminary cost estimate uses "main" parameters, i.e., parameters that have the most significant cost impact on the product being estimated. It focuses on cost drivers, the specified design and/or planning characteristics that have a predominant effect on the cost of a project. Once the cost drivers are identified, cost models for the generation of conceptual estimates can then be developed. Reliance on conceptual cost estimates generated by properly developed and carefully evaluated cost models can save the user time and resources not only in the evaluation of project alternatives but also in the checking of detailed cost estimates prior to bid submittals (CII 1998, Barrie and Paulson 1992). Therefore, new alternatives for cost prediction techniques for the early design stage are investigated in this study.

Table 2.1. AACE international cost estimation classifications

Estimate Class	Level of Project Definition (%)	End Usage (Typical Purpose)	Expected Accuracy Range (%)
Class 5	0 to 2	Concept Screening	-50 to +100
Class 4	1 to 5	Study or Feasibility	-30 to +50
Class 3	10 to 40	Budget or Control	-20 to +30
Class 2	30 to 70	Control or Bid/Tender	-15 to +20
Class 1	50 to 100	Check Estimate or Bid	-10 to +15

Table 2.2. Construction industry institute cost estimate definitions

Estimate Class	Percentage Range	Description/Methodology
Order of Magnitude	± 30 to 50	Feasibility Study: cost/capacity curves
Factored Estimate	±25 to 30	Major equipment: cost/factors
Control Estimate	±10 to 15	Quantities: mech./elec./civil drawings
Detailed or Definitive	±<10	Based on detailed drawings

2.2. Machine (Predictive) Learning

The rise in computing power has been accompanied by a rapid growth in statistical modeling and data analysis. New techniques have emerged for predictive learning, not possible 10 years ago, using ideas that bridge the gaps among statistics, computer science, and artificial intelligence (Hastie 2004). In this chapter, some of these new methods, namely artificial neural networks (ANN) and case based reasoning (CBR) are covered with emphasis on their possible application to cost prediction problems.

The predictive or machine learning problem is easy to state but difficult to solve in general. Given a set of measured values of attributes /characteristics/ properties on a object (observation) $x = (x_1, x_2,..., x_n)$ (often called "variables") the goal is to predict (estimate) the unknown value of another attribute y. The quantity y is called the "output" or "response" variable, and $x = (x_1, ..., x_n)$ are referred to as the "input" or "predictor" variables. The prediction takes the form function $y = F(x_1, x_2, ..., x_n) = F(x)$ that maps a point x in the space of all joint values of the predictor variables, to a point y in the space of response values. The goal is to produce a "good" predictive F(x).

In predictive or machine learning one uses data. A "training" data base $D = \{y_i, x_{i1}, x_{i2}, ..., x_{in}\}_1^N = \{y_{Ni}, x_i\}_1^N$ of N previously solved cases is presumed to exist for which the values of all variables (response and predictors) have been jointly measured. A "learning" procedure is applied to these data in order to extract (estimate) a good predicting function F(x). There are many commonly used learning procedures. These include linear regression, neural networks, decision trees, etc. For descriptions of a large number of such learning procedures see Hastie, et al. (2001).

2.2.1. Artificial Neural Networks (ANN)

Artificial neural networks (ANN) are an efficient exploitation of predictive (machine) learning. They have been widely used to model some of the human activities in many areas of science and engineering. Early applications of ANN in engineering go back to the late eighties (Adeli 2001). They are also currently used by various researchers for different purposes in the fields of building systems and construction (Doğan and Günaydın 2003). One of the distinct characteristics of ANN is its ability to learn from experience and examples and then to adapt to changing situations. According

to Haykin (1994), a neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the human brain in two aspects; the knowledge is acquired by the network through a learning process, and inter-neuron connection strengths known as synaptic weights are used to store the knowledge. For more detailed information about ANN see Fausett (1994) and Haykin (1994).

ANN is good for some tasks while lacking in some others. Specifically, they are good for tasks involving incomplete data sets, fuzzy or incomplete information and for highly complex and ill-defined problems, where humans usually decide on an intuitional basis (Rafiq et al. 1998, 2001). They can learn from examples and able to deal with non-linear problems. Furthermore, they exhibit robustness and fault tolerance. The tasks that ANN cannot handle effectively are those requiring high accuracy and precision, as in logic and arithmetic (Kalogirou 1999, 2001). However, they are quite efficient for the success of the design process which depends heavily on the initial guess (Mukherjee and Deshpande 1995). They have been used in diverse applications in control, robotics, pattern recognition, forecasting, medicine, power systems, manufacturing, optimization, signal processing and social/psychological sciences.

ANN operates like a 'black box' model, requiring no detailed information about the system. Instead, they learn the relationships between the input parameters and the controlled and uncontrolled variables by studying previously recorded data. The network usually consists of an input layer, some hidden layers and an output layer (see Figure 2.1). In its simple form, each single neuron is connected to other neurons of a previous layer through adaptable synaptic weights (see Figure 2.1). Knowledge is usually stored as a set of connection weights (presumably corresponding to synapse efficacy in biological neural systems). Training is the process of modifying the connection weights in some orderly fashion using a suitable learning method. The network uses a learning mode, in which an input is presented to the network along with the desired output, and the weights are adjusted so that the network attempts to produce the desired output. The weights after training contain meaningful information, whereas before training, they are random and have no meaning.

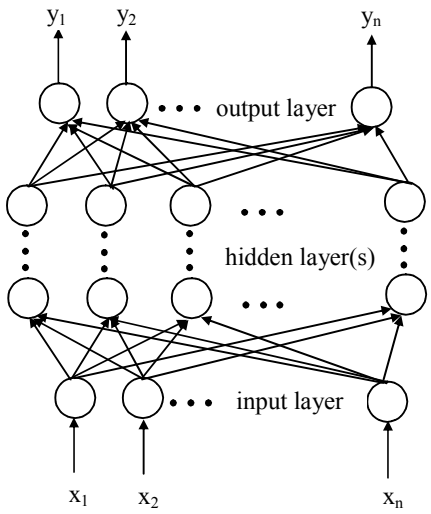


Figure 2.1. Schematic diagram of a typical multilayer feed forward neural network architecture

A single node or neuron receives weighted activation of other nodes ($x_j w_{ij}$) through its incoming connections. First, these are added (summation) (see Figure 2.2). The result is then passed through an activation function, the outcome being activation of the node (see Figure 2.2). For each of the outgoing connections, this activation value (α_i) is multiplied with the specific weight and transferred to the next node.

A training set is a group of matched input and output patterns used for training the network, usually by suitable adaptation of the synaptic weights. The outputs are the dependent variables that the network produces for the corresponding input. It is important that all the information the network needs to learn is supplied to the network as a data set. When each pattern is read, the network uses the input data to produce an output, which is then compared to the training set, i.e. the correct or desired output. If there is a difference, the connection weights are altered in such a direction that error is decreased. After the network has run through all the input patterns, if the error is still greater than the maximum desired tolerance, the ANN runs again through all the input patterns repeatedly until all the errors are within the required tolerance. When the training reaches a satisfactory level, the network holds the weights constant and uses the trained network to make decisions, or define associations in new input data sets not used to train it.

The most popular learning algorithms are the back-propagation and its variants. The back-propagation algorithm is one of the most powerful algorithms in neural networks. For further information see Rumelhart et al. (1986). The training set has to be a representative collection of input-output examples. Back-propagation training is a gradient-descent algorithm. It tries to improve the performance of the neural network by reducing the total error by changing the weights along its gradient. The error can be expressed by the mean-square value (MSE), which is calculated by:

$$MSE = \frac{\sqrt{\sum_{i=1}^{n} (x_i - E(i))^2}}{n} \qquad (2.1)$$

where n is the number examples to be evaluated in the training set, x_i is the network output (target) related to the example (i=1,2,…,n) and E(i) is the desired output. An error of zero would indicate that all the output patterns computed by the ANN perfectly match the expected values, and the network is well trained.

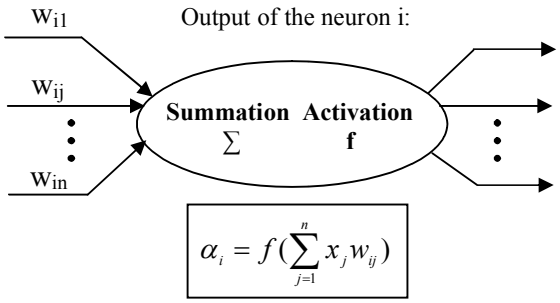


Figure 2.2. Mathematical model of an artificial neuron

According to the performance of the system, ANN model is set for future predictions with new data. There are no rules as to when an ANN technique is more or less suitable for an application and the selection of the model is done empirically after testing various alternative solutions. However, based on the work they have carried out so far, it is believed that ANN offers an alternative method for predictive learning, which should not be underestimated.

Various researchers have used neural networks as a tool for prediction and optimization previously. But in the area of cost estimating there exist only few applications. The works of Shtub and Versano (1999) and Zhang and Fuh (1998) in the manufacturing industry, comprise alternative ANN models for cost estimating. Shtub and Versano (1999) have developed a system that was based on a neural network that was trained to interpret cost estimates when a new technology was introduced for steel pipe bending. They also found out that neural networks outperform linear regression analysis for cost estimation. Zhang and Fuh (1998) designed a neural network model for early cost estimation of packaging products. In their model, they extracted and quantified cost-sensitive attributes of a product design. The correlation between these cost features and the final cost of the product was found by using a back propagation neural network algorithm depending on historical data. In the construction industry, Adeli and Wu (1998) formulated a regularization neural network to estimate highway construction costs which were very noisy. They observed that as the number of attributes was increased, the construction cost was estimated more accurately. In another study a neural network model for parametric cost estimation of highway projects was proposed by using spreadsheet simulation (Hegazy and Ayed 1998). Hegazy and Ayed (1998) developed a very adaptable and flexible model of ANN by simply facilitating a spreadsheet program. One particular study by Harding et al. (2000) constructed an ANN model, which aimed to provide an accurate comparative cost of different procurement routes. Among the 40 variables they used in their study were design specific criteria such as the frame type and gross internal floor area. Emsley et al. (2002) suggested that procurement routes cannot be isolated from cost significant variables (i.e., design and site related variables, project strategic variables) in a building project. Therefore they developed Harding et al.'s (2000) model one step further by using a more complete and sophisticated data set which would not be available at the early design stage. Their findings indicated 16.6% mean absolute percentage error.

Al-Tabtabai et al. (1999) also developed a neural network model that could be used to estimate the percentage increase in the cost of a typical highway project from a baseline reference estimate. They used environmental, company and project specific factors. Their model measured the combined effect of these factors on the percentage change in expected cost. The network generated outputs reaching a mean absolute percentage error of 8.1%. Squeira (1999b) presented an automated cost estimating system for low-rise structural steel buildings by utilizing design variables such as area, perimeter, height, load, etc. He used a commercial software of ANN and showed that the neural network model outperformed regression. The mean absolute percentage error calculated for the neural network model and regression equation over the entire data set were 11% and 15%, respectively, for the cost estimating of structural steel framing. For the two other models (i.e., total direct cost and cost of wall panels), the mean absolute percentage errors for the neural network approach and regression were 13% vs. 21% and 18% vs. 57%, respectively. Creese and Li (1995) developed a neural network application for the parametric cost estimating of timber bridges and again found that the neural network method outperformed common linear regression methods. Their study also showed that the estimation using neural networks improved when more independent variables were introduced in training. However, Bode (1998) concluded in his research report that neural networks can only work with a limited number of cost drivers, and more attributes with cost effects need larger case bases for the learning algorithm to achieve satisfying accuracy. Setyawati et al. (2002) compared their results with those of Creese and Li (1995) and pointed out the inappropriateness of standard statistical methods for cost estimating and suggested regression analysis based on percentage error and on combined methods for obtaining the appropriate linear regression which might outperform ANN models for cost estimating. Smith and Mason (1997) also examined the performance, stability, and ease of cost estimation modeling using regression versus neural networks to develop cost estimating relationships. They reported that the cost data did not enable fitting a commonly chosen model, or did not allow the analyst to discern the appropriate cost estimating relationships; the problem of model commitment became more complex as the dimensionality of the independent variable set grew.

2.2.2. Case Based Reasoning (CBR)

Case based reasoning (CBR) involves applying past experiences, in the form of prior cases, to guide current decision making. In essence, the cased based reasoner assigns an outcome to a problem based on the outcomes of the recently similar prior cases. A prior case may be a template for a solution to the problem or the basis of an argument how to decide it.

A case is considered as a set of features, attributes, and relations of a given situation and its associated outcome(s). Although the structure of a case may differ from one domain to the next, the concept of a case is the same. A case is situation-specific, unlike a rule, which is a unit of generalized knowledge (Gupta 1994).

Essentially the roots of case-based reasoning in AI are found in the works of Schank (1982) on dynamic memory and the fundamental role that a reminding of earlier situations have in problem solving and learning. For a bibliographic categorisation and review of CBR research see Aamodt and Plaza 1994, Watson and Marir 1994a, 1994b.

CBR development consists of four steps. The first is to design the structure of the case. The second is to collect cases. The third is to prototype the similarity retrieval. Finally, the prototype undergoes successive refinement. The processes involved in CBR can be represented by a schematic cycle (see Figure 2.3). Aamodt and Plaza (1994) have described CBR typically as a cyclical process comprising the four REs:

1. RETRIEVE the most similar case(s);
2. REUSE the case(s) to attempt to solve the problem;
3. REVISE the proposed solution if necessary, and
4. RETAIN the new solution as a part of a new case.

A new problem is matched against cases in the case base and one or more similar cases are *retrieved*. A solution suggested by the matching cases is then *reused* and tested for success. Unless the retrieved case is a close match the solution will probably have to be *revised* producing a new case that can be *retained*. This cycle currently rarely occurs without human intervention. For example many CBR tools act primarily as case retrieval and reuse systems. Case revision (i.e., adaptation) often being undertaken by managers of the case base. Well known methods for case retrieval are: nearest neighbour, induction, knowledge guided induction and template retrieval. These methods can be used alone or combined into hybrid retrieval strategies. For a further

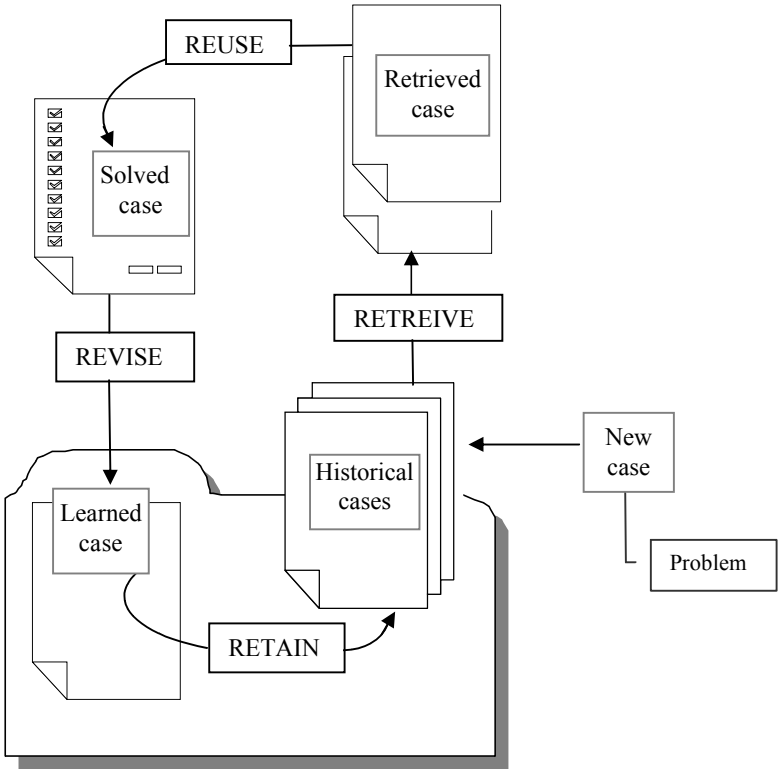


Figure 2.3. Basic CBR approach

overview of the theoretical principles of CBR, see Kolodner (1991, 1993), Riesbeck and Schank (1989).

CBR systems handle missing data well. If the current situation is missing the value of an important feature, that feature is simply not used during similarity retrieval. The most similar cases will match well with target case, except for the missing feature. The retrieved cases will possess a variety of values for the missing feature so its influence and importance can be determined. If the outcomes represented in the retrieved cases are similar, the missing feature is unimportant, and a prediction can be made with confidence. If the outcomes vary widely, the prediction should be delayed until the value of the missing feature is determined for the target case.

One of the important strengths of CBR that sets it apart from most AI techniques is that a CBR system is aware of its own limitations. If no similar cases are retrieved, the CBR system cannot make a prediction. This process is far superior to making a nonsensical prediction as most systems would. However, the biggest weakness of CBR is its requirement for cases. Enough cases should be present in the case base so that a similar one is retrieved. The sparser the case base, the more effort must be invested into adaptation strategy. In the extreme, making a prediction from a case that is not very similar to the current situation is just as difficult as making it from scratch. It's not so much the absolute number of cases in the case that is important as the density of cases in the case base. In some domains, few cases are required to fill the case base to the required density. In other domains, especially those with many important features, a very large number may be required.

Prediction is a universal problem in industry. Case-based reasoning (CBR) can be a good solution to prediction problems. The number of rules required to generate a cost prediction, taking into account all relevant variables, is usually quite large and time consuming to generate. Typically, this knowledge must include how to decompose the project into smaller tasks and accurately estimate the cost of each portion. Therefore, the knowledge required to predict the costs from scratch is enormous. CBR avoids this knowledge-acquisition bottleneck by using the wealth of existing prediction knowledge embodied in past cases (Stottler 1994). Since the prediction is not generated from scratch but is adjusted from a previous experience, less specific and less accurate knowledge is required. Yau and Yang (1998) presents an example how CBR can be used to estimate construction duration and costs of building construction projects at the preliminary design stage. Neural networks also make use of past experience (in the form

of training data), but they cannot easily justify the prediction they make. CBR systems can point to the similar cases on which the prediction is based as justification. Any required adjustments from these cases are usually small and therefore credible. In addition, since the knowledge is in symbolic form, richer meaning can be conveyed. For example, an architectural design project might have gone over budget because of numerous change requests from the client. A text document describing this reason can be stored along with the case. Later, when this project is retrieved as a similar case to estimate design costs for current projects with the same client, the architect is warned about the nature of the excessive project cost in addition to the higher than usual cost prediction. Therefore, the architect can explain to the client the reason for the higher cost or include a maximum number of change requests in the contract.

CBR's ability to mimic the decision-making processes of humans provides an alternative in solving experience-oriented problems when traditional techniques or ANN encounter difficulties.

CHAPTER 3

METHODOLOGY

The construction industry utilizes experience; therefore knowledge and appreciation of previous experience are critical to resolving problems that may reoccur. Artificial neural networks (ANN) and case based reasoning (CBR) have grown to be effective techniques in the machine learning domain that offer alternatives for solving construction related problems that require extensive experience. Late literature reviews demonstrated the potential benefits of these techniques in construction management and its superior performance over other traditional prediction techniques (Arditi and Tokdemir 1999a, Arditi and Tokdemir 1999b, Günaydın and Doğan 2004, Hegazy and Ayed 1998, Yau and Yang 1998). Further exploring ANN's and CBR's capabilities in the construction management domain is a worthwhile task. Recent research studies about the effectiveness of integrated machine learning approaches indicate that these systems could achieve better results when enhanced by other techniques (Cardie 1993, Jarmulak and Craw 1999, Jarmulak et al. 2000, Ling et al. 1997, Shin and Han 1999).

In order to reach the previously stated goals of this study, ANN and CBR models are developed, enhanced and tested for providing better tools of cost prediction at the early design stage. Therefore, data pertaining to the early design parameters and unit cost of the superstructure of residential building projects are used to test developed ANN and CBR models.

Hegazy and Ayed's (1998) spreadsheet-based ANN prediction model and a commercial software NeuroSolutions (2002) are used for the ANN modeling. In order to determine ANN weights, as an alternative to back-propagation training of NeuroSolutions, two other techniques, namely simplex optimization, and genetic algorithms (GA) are used. Simplex optimization and GA are implemented using Excel add-in programs Solver and Evolver (1998), respectively.

The CBR prediction model is established by developing an Excel-based simulation. The model is assigned attribute weights by six different weight generation methods in order to compare their impact on the performance of CBR prediction. The weights for the attributes of the CBR-Excel model are computed by (1) the feature

counting method, (2) the gradient descent method, (3) genetic algorithms (GA), (4) the binary-dtree method, (5) the info-top method, and (6) the info-dtree method.

Three commercial software help to determine weights in the CBR-Excel model. A CBR software called Esteem (1996) is used to implement the gradient descent weight generation method. Evolver (1998) is used once more for GA computations. Binary-dtree, info-top and info-dtree methods named by Ling et al. (1997) are adapted by using induction decision trees (ID3). The decision tree of the cost prediction problem is constructed by using the See5 software (1997).

Boosted decision trees (BDT) constructed by using See5 is used as the third machine learning technique, as an alternative to ANN and CBR models, for the cost prediction problem at hand. However, cost data had to be classified into a large number of classes because of the BDT modeling rules. The small number of cases available in this study produced less accurate outcomes than the ANN and CBR outcomes. The BDT model and the results achieved are presented in Appendix B.

The development of the ANN and CBR models, and the weight generation methods are described in this chapter. The results are discussed, ANN and CBR techniques are compared, and concluding remarks are made in the next chapter. The methodology of the study is presented in Figure 1.1.

3.1. Spreadsheet Simulation of ANN

In this section, a spreadsheet simulation model of a three-layer ANN (Figure 3.1) with one output node is presented on Microsoft Excel. Many practitioners are familiar with spreadsheet applications. As a simple and more transparent approach to ANN modeling, Excel based simulation is adapted from Hegazy and Ayed's (1998) study. The spreadsheet represents a template for one hidden-layer ANN that is suitable for most applications (Hegazy et al. 1994). The processing of the template incorporates seven steps, following the widely known back-propagation formulation (Rumelhart et al. 1986). The general structure and computations of this type of ANN are presented in the following steps:

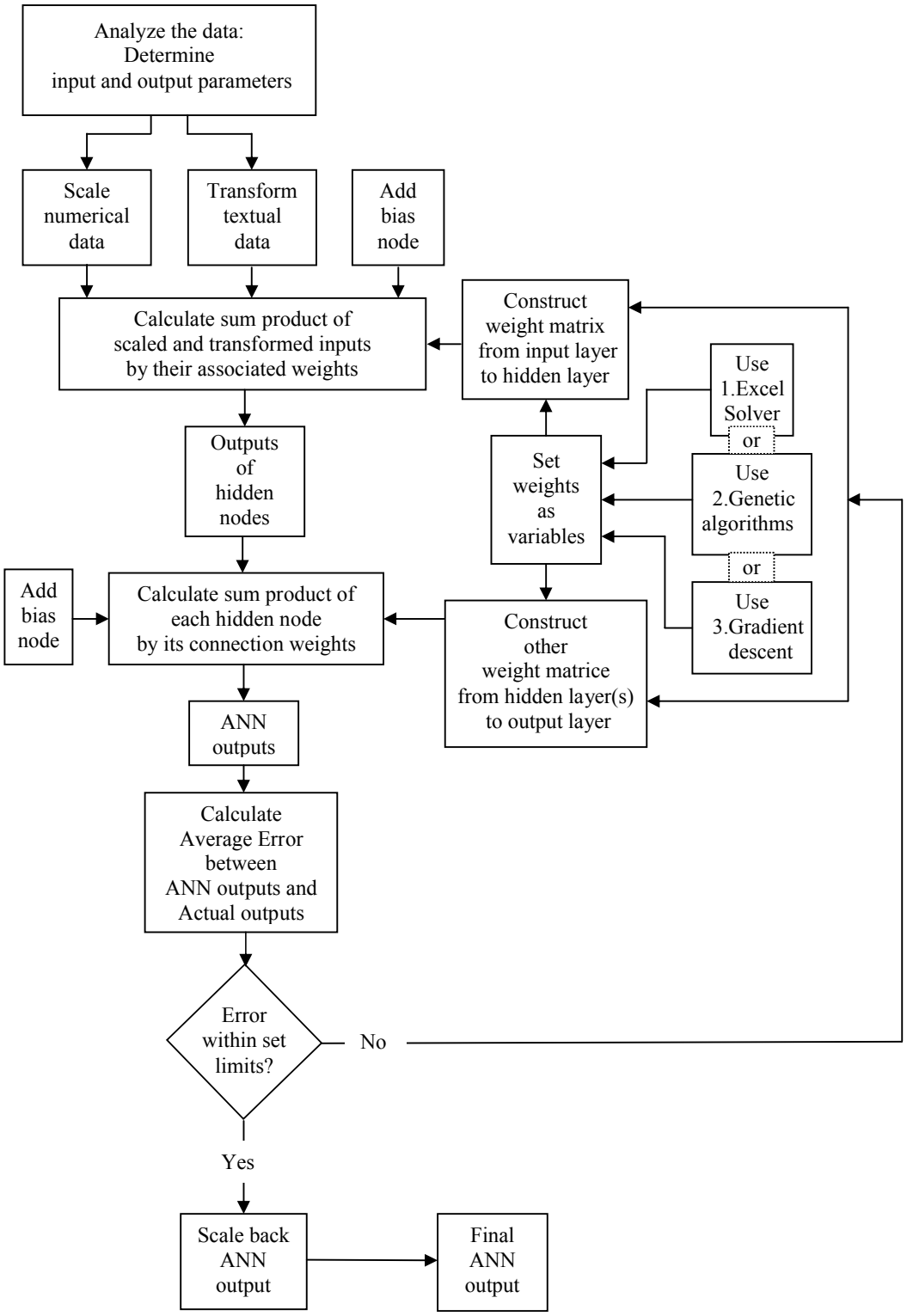


Figure 3.1. Basic process of ANN

Step 1. *Data Organization* – as a preliminary stage to ANN modeling, the problem at hand needs to be thoroughly analyzed. Through this process, the independent factors affecting the problem are identified and considered as (*N*) input parameters represented by nodes at the input layer of an ANN. Similarly, the number of associated outputs or conclusions (*O*) are represented by nodes at the output layer. Once input and output parameters are identified their corresponding data are collected from the (*P*) case studies. These data become available for the training stage of the ANN. Schematic illustration of ANN Excel simulation notations of N, P, O, etc. are shown in Figure 3.2.

To implement this step in an Excel spreadsheet, the data is first transformed into numerical values and stored in a data-list that is a matrix of (*N*+*O*) columns and (*P*) rows (Figure 3.3). The numerical transformation of textual data can be done in either a continuous or binary manner. In continuous transformation the value of a parameter called 'season', for example, can be an integer 0-3 for winter, spring, summer, and fall respectively. Alternatively, in a binary transformation, four parameters are used to represent the four seasons and only one of them is assigned a value 1, whereas the others are 0. Depending on the type of transformation used, the number of ANN nodes (*N*) will be determined and, accordingly, the size of the spreadsheet matrix. For each variable, the minimum and maximum values were also put in spreadsheet formulas to be used in Step 2.

Step 2. *Data Scaling* – In this step, the input-data part of the first matrix (*N* columns by *P* rows) is scaled to a range from [-1 to 1] to suit NN processing. This is done by constructing a second matrix with a linear formula for scaling the values of the first matrix, as follows:

$$Scaled\ Value = \frac{2 \times (UnscaledValue - ColumnMin)}{(ColumnMax - ColumnMin)} - 1 \qquad (4.1.)$$

This scaling formula is written in only one cell (B15 for example, in Figure 3.4), and then copied to all cells in the scaling matrix. To the right of this matrix, a column was added with unit values associated with the bias node, as illustrated in Figure 3.4.

Step 3. *Weight Matrix (W)* – the third step is to construct and initialize the weight matrix between the inputs and the hidden layer (Figure 3.5). All inputs (1 to *N*) and a bias node

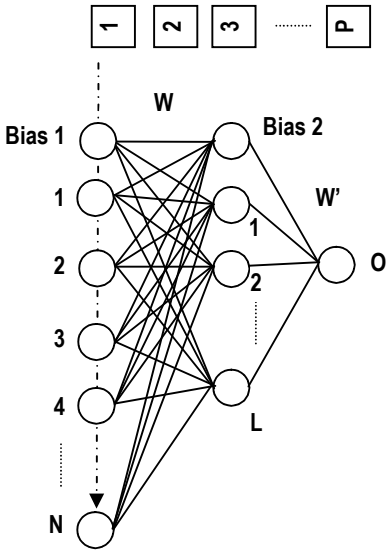


Figure 3.2. Schematic illustration of ANN Excel simulation notations of N, L, O, P, W, W'

	A	B	C	D	E	F
1	Project No.	Inputs				Outputs
2		1	2	…	N	O
3	1					
4	2					
5	3					
6	⋮					
7	P					
8	Min.:	=MIN(B3:B7)				=MIN(F3:F7)
9	Max.:	=MAX(B3:B7)				=MAX(F3:F7)

Figure 3.3. Step 1: Organization of row data

	A	B	C	D	E	F
⋮						
13	Project No.	Scaled Inputs				
14		1	2	…	N	Bias 1
15	1					1
16	2					1
17	⋮					1
18	P					1

=2*(B3-B$8)/(B$9-B$8)-1
Made once and copied to all cells

Figure 3.4. Scaling of input values to a range (-1,1)

	A	B	C	D	E	F
⋮						
25	To Hidden	Weights from Inputs & Bias 1				
26		I'_1	I'_2	…	I'_N	Bias 1
27	Node 1					
28	Node 2					
29	Node 3					
30	⋮					
31	Node L					

Cells contain weights values put initially as 1.0s. The matrix elements are set as variables in the optimization.

Figure 3.5. Weight matrix (W) from (N) inputs to (L) hidden nodes

were fully connected to the hidden nodes. The number of hidden nodes (L) was set as one-half of the total input and output nodes, as heuristically suggested in the literature (Hegazy et al. 1994). All of the values in the weight matrix are considered variables to be determined in the ANN modeling. Hegazy and Ayed (1998) suggest that setting the initial weight values to 1 is appropriate for inputs scaled to a range (-1 to 1).

Once the Excel template has been set up with initial weights of 1s, the overall performance indicator (cell D94 which will be mentioned in step 7 and Figure 3.11) was showing a very high error value. Because all formulas in the template are functions of the weights, the next step was to determine the ANN weight values that would optimize ANN performance. Three approaches were used: (1) simplex optimization using Microsoft Excel Solver. (Excel 2003); (2) GA using Evolver software from the Palisade Corporation (Evolver 1998); and (3) back-propagation training using NeuroSolutions (2002) software from the NeuroDimensions Inc.

Simplex Optimization: A simplex optimization is implemented, using Solver, an Excel add-in program. The implementation, therefore, is conducted directly on the NN spreadsheet. Solver is a powerful and easy to use optimization tool that is highly integrated with Excel. Solver can find the optimum set of values for some variables so as to maximize or minimize a target cell (or objective function) that is linked by formulas to the variables, under a set of user-specified constraints. It proceeds by first finding a feasible solution, and then seeking to improve upon it; changing the variables to move from one feasible solution to another until the objective function has reached its maximum or minimum. For the ANN (or NN) simulation described previously, Solver optimization options are shown in Figure 3.6 (solver optimization screen). The optimization objective is to minimize the NN weighted error (see Step 7, cell D94 of Figure 3.11). Also, the optimization variables, representing the adjustable cells are the weights from inputs to hidden nodes and from hidden nodes to outputs. To avoid incorrect network results on individual training cases, optimization constraints are set to limit the percentage error on the training and test cases to 2 and 5% or lower, respectively. Cell references for the optimization variables and the constraints are shown in Figure 3.6. These values give more emphasis on the test cases, similar to what is also done with back-propagation training (which will be mentioned in the following sections).

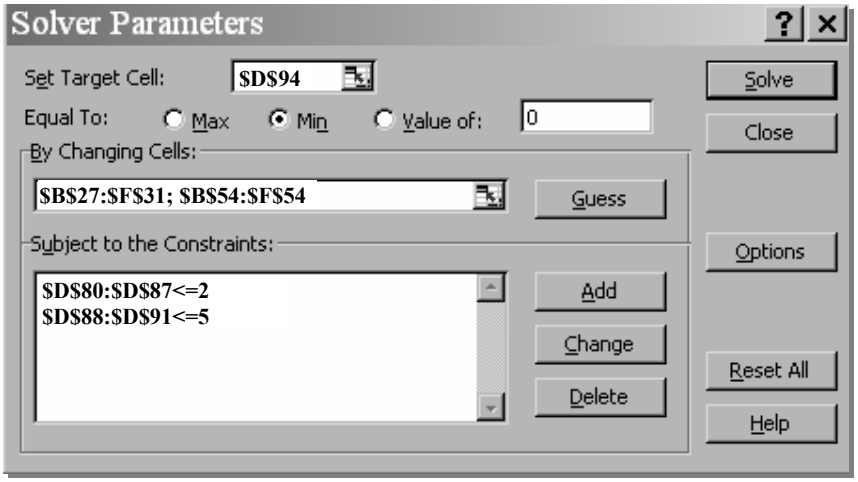


Figure 3.6. Solver optimization screen

Once the optimization parameters are input solver, the optimization process is started. Experimenting with this approach, it is found that the results are often sensitive to the initial values of the variables and some manipulation of Solver options may become necessary to arrive at the optimum solution. Using the suggested (0-1) range for the weights can be a good start. Also, when optimization is not improved over a long period of time, it can be manually stopped and then continued after reinitializing some of the weight values. Generally, the time taken by Solver to arrive at the optimum solution varies significantly depending on the size and complexity of the model. The optimization process may need to be frequently interrupted to change solver options that are fully described in Excel documentation (Excel 2003).

Genetic Algorithms: GA technique is another optimization method that is fundamentally different from traditional simplex-based algorithms such as the one used by Excel Solver. It uses the method of evolution, specifically survival of the fittest. The theory behind GA is that a population of certain species will, after many generations of random evolution, adapt to live better in its environment. GA solves optimization problems in the same fashion. First, a population of possible solutions to the problem is created. Individuals in the population are then allowed to randomly breed, a process called crossover, until the fittest offspring (the one that solves the problem best) is generated (Hegazy et al. 1994). After a large number of generations, a population eventually emerges where the individuals will provide an optimum or close to optimum solution. For the case study at hand, a commercial GA software (Evolver 2004) was used to find the optimum weights of the model. Similar to Solver, Evolver works as an add-in to Microsoft Excel, and can replace Excel Solver for optimizing complicated problems. The Evolver screen is shown in Figure 3.7 with all the cell references to the optimization objective function and constraints. Similar to Solver optimization, cell D94 representing the NN weighted error is selected to be minimized. The adjustable cells containing the optimization variables (called chromosomes in GA terminology) are also specified as the two weight matrices. Optimization constraints are then set. These constraints limit the range of weight values that Evolver searches for, thus reducing processing time. In addition, the constraints add sub goals to the original objective function to limit the percentage incorrect training and test sets to 2 and 5%, respectively. During the GA optimization, Evolver options can be used to enhance the results. For example, “population size” affects processing time because the fitness function must be calculated for every individual in every generation. A population size of 50 is generally

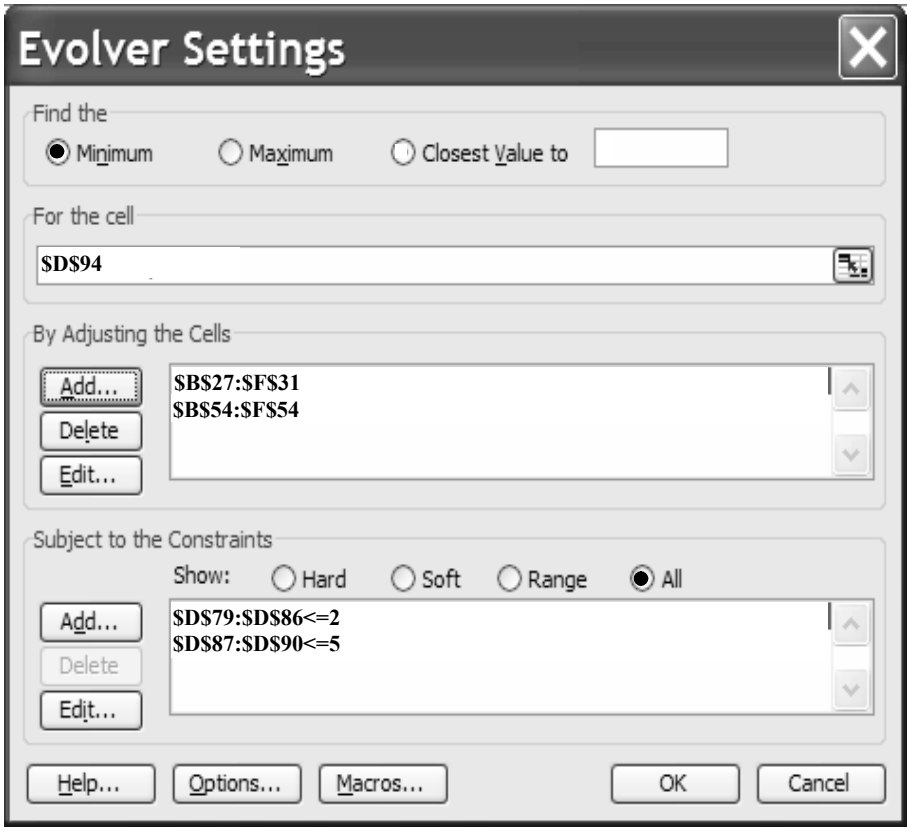


Figure 3.7. Evolver optimization screen for ANN Excel-Simulation

found as a good number to start with. This number can be increased later during the optimization process. Chromosome length presents the level of accuracy needed for the adjustable cells. More bits mean high-precision answers. Other program specific parameters that are fully described in the documentation (Evolver 2004) can be used to speed the solutions and prevent standstill progress.

Back-propagation: Back-propagation training is one of the most common methods for training NN given historical data (Rumelhart et al. 1986). In essence, back-propagation training adapts a gradient-descent approach of adjusting the ANN weights. To implement back-propagation training, a commercial NN software "NeuroSolutions" is used as a stand-alone environment for NN development. NeuroSolutions (2002) is used for its ease of use, speed of training, and its host of NN architectures, with flexible user-optimization of training parameters. In essence, back-propagation training adapts a gradient-descent approach of adjusting the NN weights. During training, an NN is presented with the data of thousands of times (called cycles). After each cycle, the error between the NN outputs and the actual outputs are propagated backward to adjust the weights in a manner that is mathematically guaranteed to converge (Rumelhart et al. 1986). To achieve good generalization, NeuroSolutions optimizes training by exposing the network to the amount of training that minimizes the average error between actual and predicted results for a group of test cases. The NN is saved whenever a new minimum average error is reached. Using the optimization features of NeuroSolutions, several training experiments can be conducted to arrive the best-trained NN. In these experiments, network parameters such as the number of hidden layers, the number of hidden nodes, network connections, and transfer functions are changed on a trial and error basis and the best result is documented. After training, the NN predictions are compared with the actual results.

Step 4. *Output of Hidden Nodes* – This step is to allow the hidden nodes to process the input data and produce values to be forwarded to the next layer. According to NN processing (reviewed in chapter 2), each hidden node *j* receives an activation X_j, which is the sum product of scaled inputs by their associated connection weights. Accordingly, each hidden node produces an output X'_j that is a function of its activation, as follows:

$$X_j = \sum_{i=1}^{N} \left(I_i \times W_{ij} \right) + B_{1j} \times 1.0 \tag{4.2}$$

$$X'_j = \tanh(X_j) \tag{4.3}$$

Experimenting with different activation functions such as linear, logistic, and tanh has shown that the tanh function produces the best results. As shown in step 4 in Figure 3.8, a formula was written for the first row of all hidden nodes and then copied to the down cells.

Step 5. *Weight Matrix (W')* – Similar to the weight matrix constructed in Step 3, a second matrix was constructed to connect the (L) hidden and bias nodes to the single output node (Figure 3.9). These weights are additional variables in the model and were initialized as previously described.

Step 6. *Final ANN Output* - Similar to Step 4, the output of the ANN (O) is computed by calculating the sum product (Y) of each hidden node by its connection weight and then processing this value through the tanh function as follows (see Figure 3.10 for Excel calculations):

$$Y = (\sum_{j=1}^{L} X'_j \times W'_{j1}) + B_2 \times 1.0 \tag{4.4}$$

$$O = \tanh(Y) \tag{4.5}$$

Step 7. *Scaling Back NN Output and Calculating the Error* – In this step, the NN output (O) is scaled back to the original range of value using the reverse of formula (4.1) as follows:

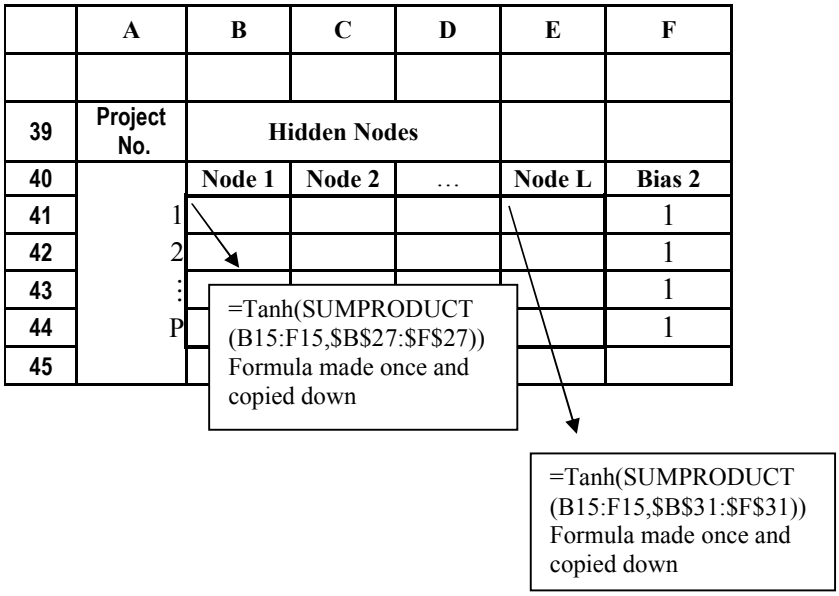


Figure 3.8. Outputs of hidden nodes

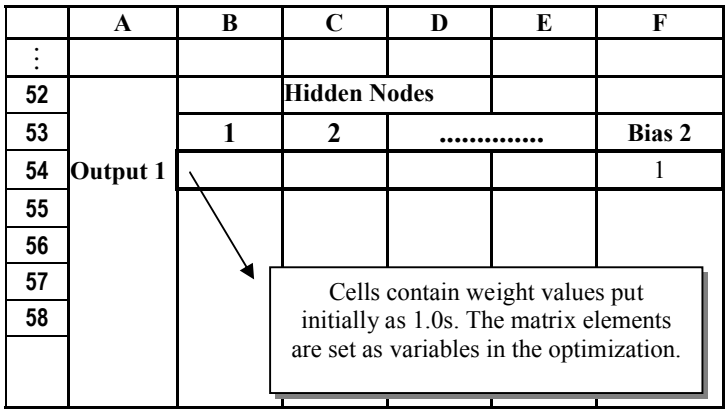


Figure 3.9. Weights W' from hidden nodes to output nodes

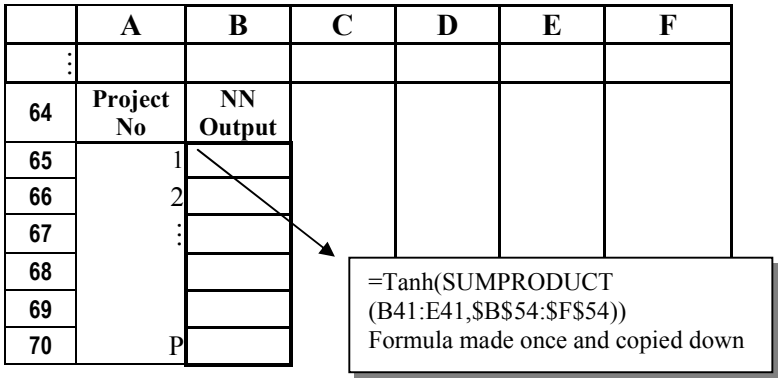


Figure 3.10. Final NN Outputs

Output Scaled Back

$$= \frac{(Output\ Value + 1)(Max\ Output - Min\ Output)}{2} + Min\ Output \tag{4.6}$$

To calculate a measure of the ANN performance, a column is constructed (Figure 3.11) for determining the error between the actual output and ANN output as follows:

$$Estimating\ Error\ (\%) = \frac{(Neuralnetwork\ Output - Actual\ Output)}{Actual\ Output} \times 100 \tag{4.7}$$

It is also possible in the ANN simulation to use some cases for training and others for testing. The average error of each group of cases can be calculated in a different cell and then combined in a cell that calculates the performance measure of the ANN, for example:

Weighted Error (%) = 0.5 (Test Set Average Error) + 0.5 (Training Set Average Error),

where weights of 0.5 and 0.5 are assumed for illustration. This approach gives more emphasis to the test cases (which are usually a small number as compared to training cases), to ensure good generalization performance and avoid overtraining.

3.2. Spreadsheet Simulation of CBR

In this section, a CBR model (Figure 3.12) is developed and simulated in a spreadsheet format and the model is set up in Microsoft Excel. This spreadsheet model represents a template for many prediction problems. The processing of the template involves six steps:

Step 1. *Organizing and Formatting Data* – The data are organized in the form of two matrices, one for the test cases and one for the input cases such as those presented in Figure 3.13. Around 10% of all cases can be designated as test cases. The input and test cases are represented in rows and the input attributes are represented in columns. The output attribute is placed in a column next to the input attributes. The values of the

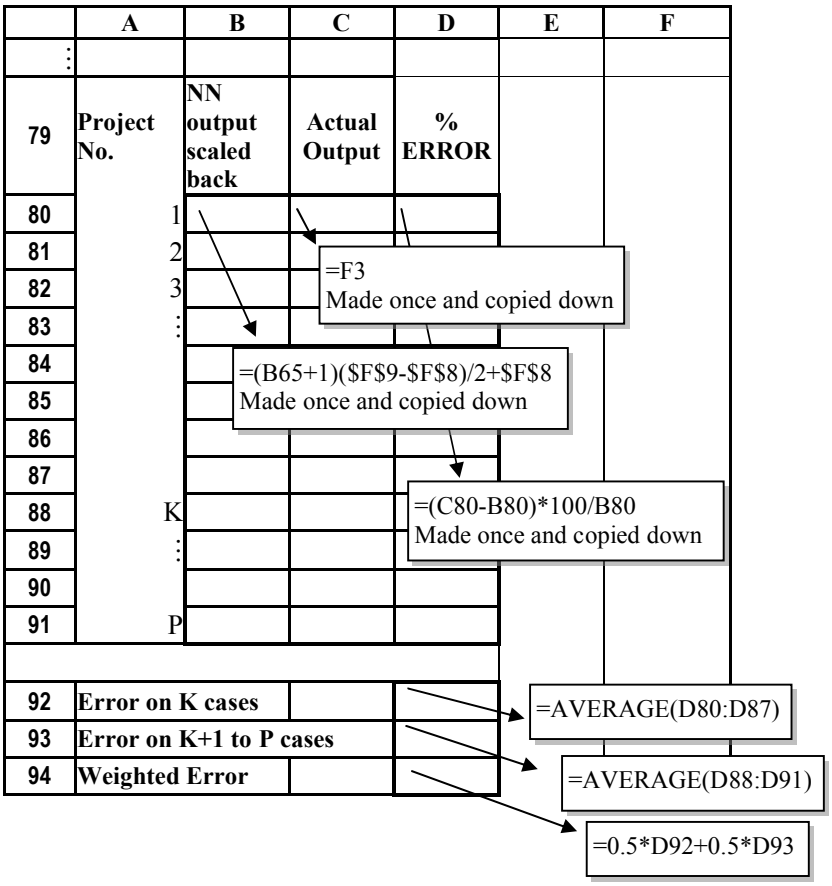


Figure 3.11. Scaling output back & calculating the error

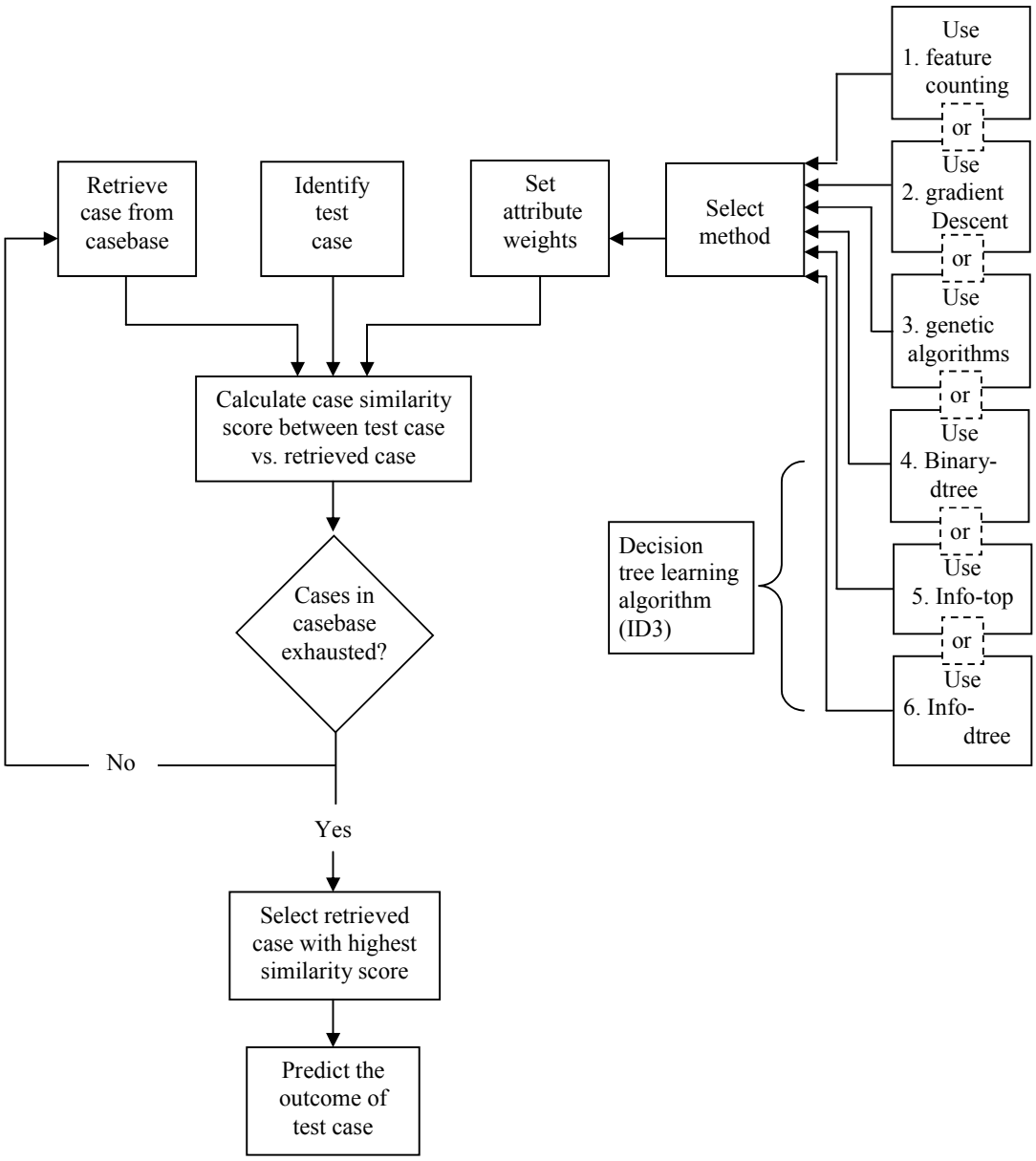


Figure 3.12. Basic process of CBR

1	A	B	C	D	E	F	G	H
2	Weights	w_1	w_2	w_3	…		w_p	0
3	**Case No.**	TEST CASEBASE **Attributes**						**Output Attribute**
4		1	2	3	…		p	
5	Case 1	I_{11}	I_{12}	I_{13}	⋯		I_{1p}	O_1
6	Case 2	I_{21}	I_{22}	I_{23}	⋯		⋮	O_2
7	⋮	⋮						⋮
8	Case m	I_{m1}	I_{m2}	I_{m3}	…		I_{mp}	O_m
9								
10	**Case No.**	INPUT CASEBASE **Attributes**						**Output Attribute**
11		1	2	3	⋯		p	
12	Case 1	I'_{11}	I'_{12}	I'_{13}	⋯		I'_{1p}	O'_1
13	Case 2	I'_{21}	I'_{22}	I'_{23}	⋯		⋮	O'_2
14	⋮	⋮						⋮
15								
16	Case n	I'_{n1}	I'_{n2}	I'_{n3}	…		I'_{np}	O'_n
17								

Figure 3.13. Formatting data to a case spreadsheet

attributes for each test and input case are represented respectively by I_{ik} and I'_{jk} where I_{ik} represents the value of attribute k (k = 1, 2, … , p) for test case i (i = 1, 2, … , m), and I'_{jk} represents the same type of information for input cases j (j = 1, 2, … , n). The weights of the attributes w_k (k = 1, 2, … , p) are located at the top of the matrix in a row that corresponds to individual attributes. The way these weights are set is explained in Step 3. After formatting, semantic information is added to the data in the form of numerical and textual attribute values.

Step 2. *Calculating Attribute Similarities* – Attribute similarity functions are used to define how similar the attribute values are to each other. Attribute similarities are computed with respect to each test case versus every case retrieved from the input casebase. Examples of textual and numerical similarity calculations are presented in Figure 3.14. Attribute similarity is denoted by S_{ijk} where i is the test case $(i = 1, 2, ..., m)$, j the input case $(j = 1, 2, ..., n)$ and k the attribute $(k = 1, 2, ..., p)$.

Assuming that the value of the first attribute for the first test case I_{11}, (in cell B5 in Figure 3.13) is textual, its similarity with the corresponding attribute value I'_{11} (in cell B12 in Figure 3.13) is established as follows:

If text in I_{11} appears to be exactly the same as text in I'_{11}, *then* similarity $S_{111} = 1$, *or else* similarity $S_{111} = 0$ (See Figure 3.14 for spreadsheet calculations) (4.8)

Assuming that the value of the third attribute for the first test case I_{13} (in cell D5 in Figure 3.13) is numerical, its similarity with attribute value I'_{13} in the corresponding cell (D12 in Figure 3.13) is established as follows:

$$S_{113} = \frac{\min\left(|I_{13}|, |I'_{13}|\right)}{\max\left(|I_{13}|, |I'_{13}|\right)}$$ (See Figure 3.14 for spreadsheet calculations) (4.9)

Step 3. *Establishing Attribute Weights* – After all the attribute similarity values are calculated in (n × p) matrices, once for each test case (the matrix for Test Case 1 is presented in Figure 3.14), the next step is to construct the weight vector that will be used in computing case similarities.

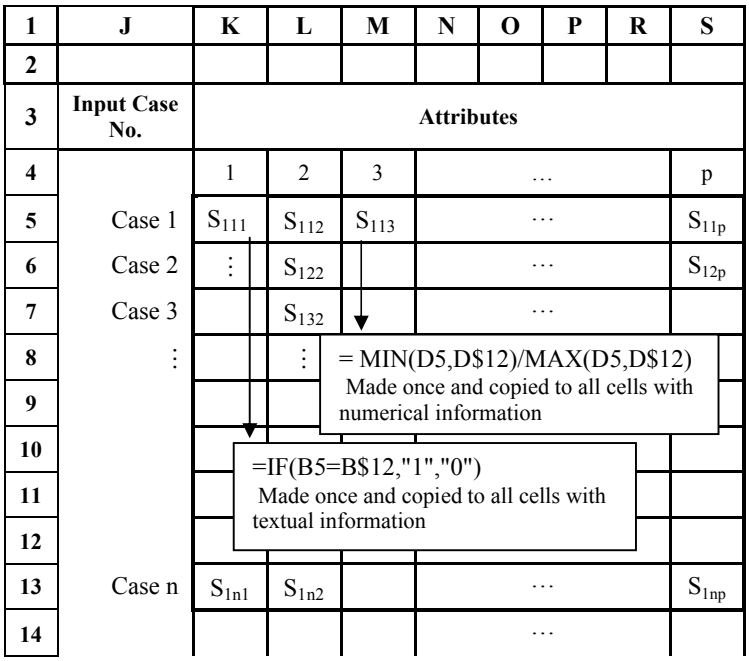


Figure 3.14. Attribute similarity matrix for Test Case 1 (i = 1).
(m similar matrices are generated, one for each test case)

Weights assign a value of importance to each attribute. In general, retrieval of the most relevant case is determined by the presence of a greater number of higher priority (more important) attributes matching between the test case and the retrieved case.

In this CBR study, weights for attributes were computed by (1) the feature counting method, (2) the gradient descent method, (3) genetic algorithms (GA), (4) binary-dtree method, (5) info-top method, and (6) info-dtree method.

Feature counting method: In the feature counting method, the weight of each input attribute is entered as 1 into the CBR Excel model, implying that attributes have equal importance (Esteem 1994). In the absence of specific information, it is assumed that there is no reason for an attribute to be more important than another.

Gradient descent method: A CBR software called "ESTEEM" was used to implement the gradient descent weight generation method. The gradient descent weight generation method's basic algorithm is presented in Figure 3.15. Random cases are selected from the input casebase, and the cases that are most similar to them (based on the initially set weights of the attributes) are found. Information on how much the weights of the attributes should be incremented or decremented is calculated considering these cases, based on how well the cases' outputs match. After examining several random cases, the resulting weights are adjusted by using a factor Delta. The factor Delta is then decreased, and the algorithm begins examining more random cases. This process continues until Delta reaches a minimum value specified by the user. When the "arithmetic" method is chosen, Delta is decremented by some value (which must be between 0 and 1) every iteration. When the "geometric" method is chosen, Delta is multiplied by some factor (which must be between 0 and 1) every iteration. The user must also specify the starting and the final value of Delta, the number of random cases that are examined every iteration, and an update parameter that specifies how quickly Delta decreases from iteration to iteration. All parameters have default values which were used in this study: the "geometric" method was used with the starting and ending values of 0.5 and 0.02 for Delta, respectively; the update parameter and the number of cases to be tested per each iteration were taken as 0.9 and 5, respectively. The weights generated by the gradient descent method were plugged into the CBR Excel model manually.

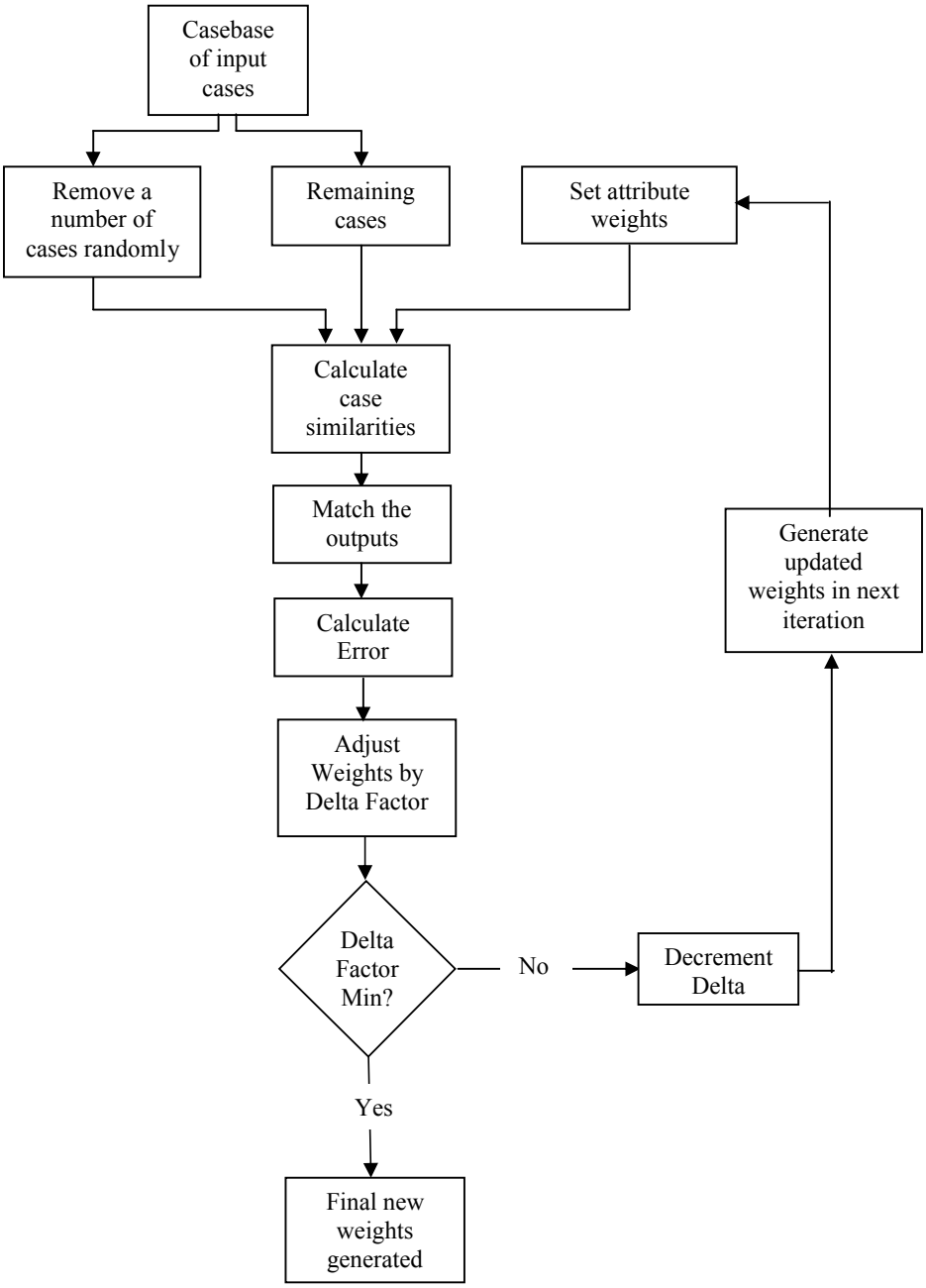


Figure 3.15. Using gradient descent to optimize CBR weights

GA method: GA uses the method of evolution, specifically "survival of the fittest." The theory behind GA is that a population of certain species will adapt to live better in its environment after many generations of random evolution. Thus, GA first creates a population of possible solutions to the problem. Individuals in the population are then allowed to randomly breed, which is called crossover, until the fittest offspring (the one that solves the problem best) is generated. After a large number of generations, a population eventually emerges where the individuals will provide an optimum solution. For this study, a commercial GA software, Evolver, was used to find the optimum weights of the model. Evolver works as an add-in to Microsoft Excel (Evolver 1998). Weights generated by Evolver were plugged into the CBR Excel model manually.

Figure 3.16 shows the flowchart of the GA optimization process used in this study. In order to use GA to generate weights, one of the cases in the input casebase is removed and called an "evaluation case." The similarities between the attributes of the evaluation case and the corresponding attributes of the remaining cases are calculated by using Equations 4.8 and 4.9. Given the start-up assumption that attributes have equal importance, case similarities (CS) are derived between the "evaluation case" versus the remaining input cases by taking the average of all attribute similarities. The relationship that governs the similarity (CS) of the input case that has an output that is closest to the output of the evaluation case is plugged into the GA algorithm (Evolver) for maximization (for taking it closer to 1).

The Evolver optimization screen is shown in Figure 3.17 with the adjustable cells containing the optimization variables (called attribute weights in the CBR system and chromosomes in GA terminology). In this study, the range of the attribute weights was set between 1 and 10, the default population size of 50 was used, and Evolver was run 15,000 times to find the optimum attribute weights that generated the maximum case similarity CS (closest to 1). This process was repeated as many times as the number of cases in the input casebase by taking a different case out as the "evaluation case" at each cycle. The averages of the weights produced by GA at each cycle were used to run CBR in Step 4.

In the remaining three other weight determining methods for attributes in the CBR simulation study, decision tree learning algorithms (ID3) are used. Related information about decision trees can be found in Cardie (1993) and Danyluk (2004).

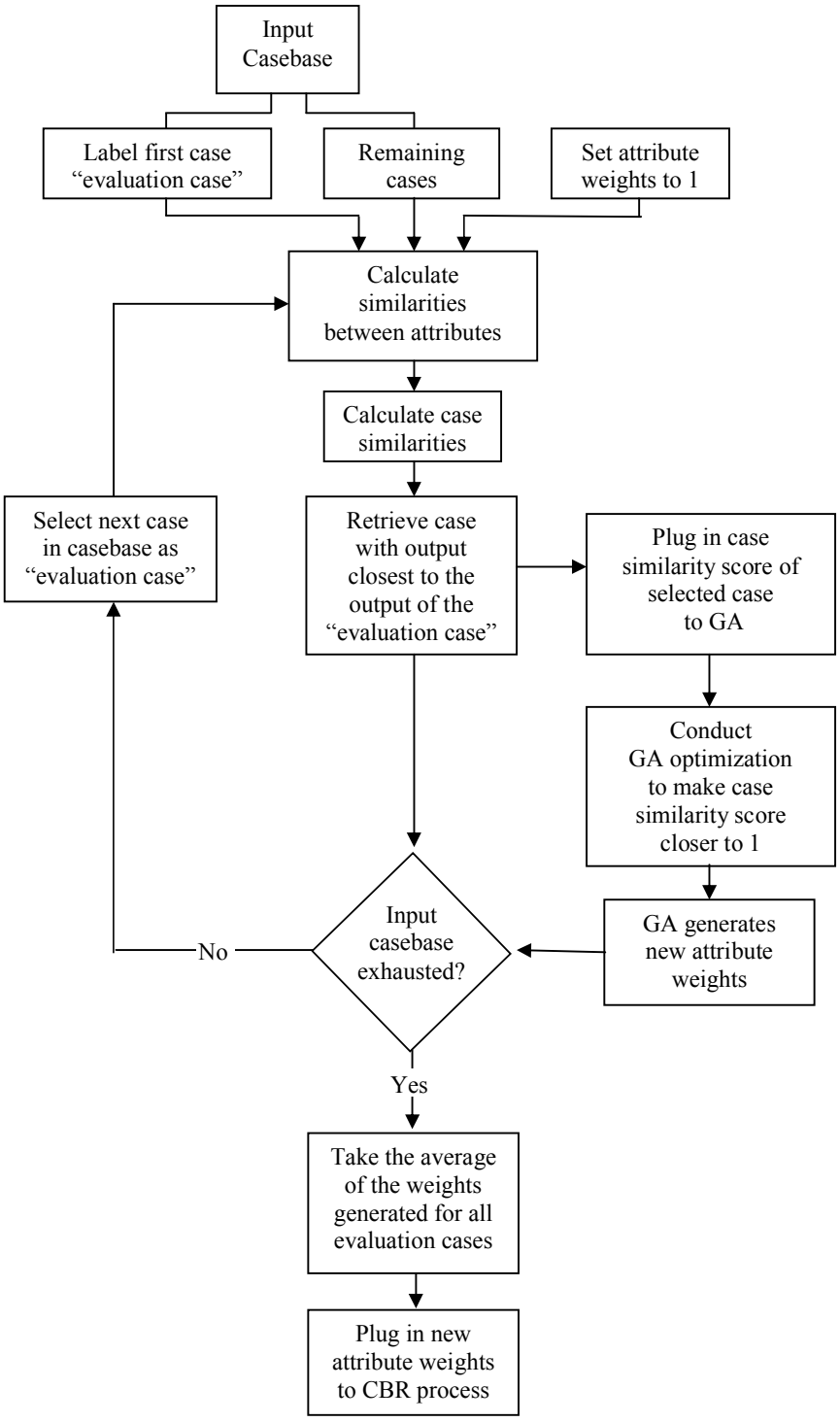


Figure 3.16. Using GA to optimize CBR feature weights

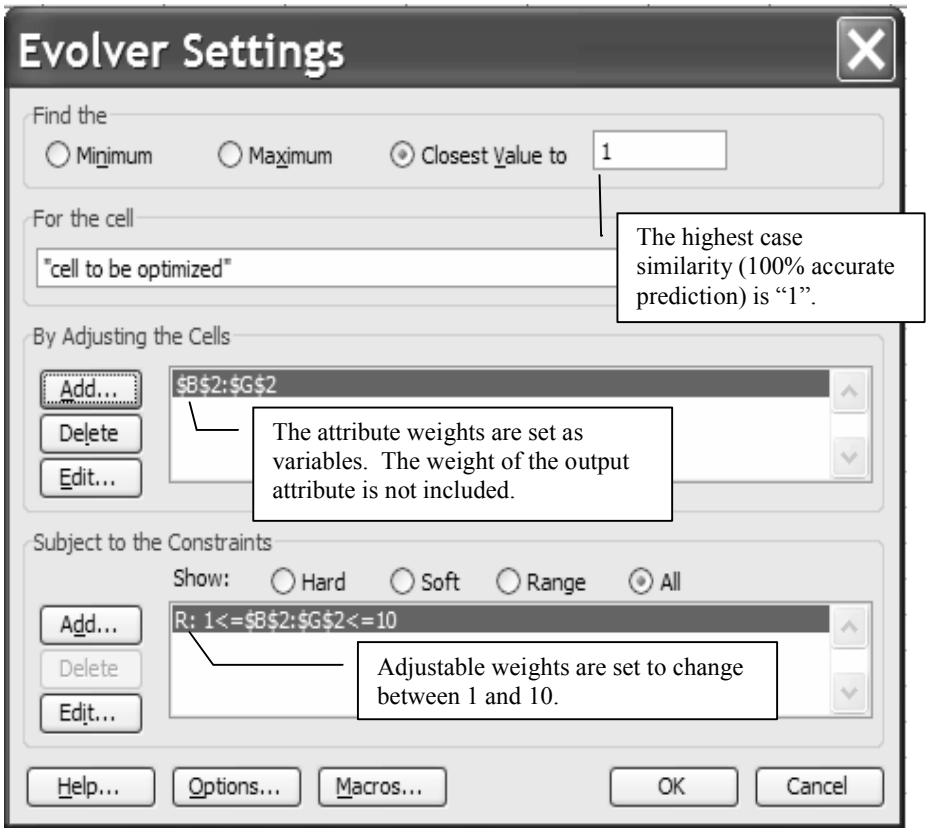


Figure 3.17. Evolver optimization screen for optimization of CBR attribute weights

Decision Trees: A decision tree is a tree in which each branch node represents an attribute, and the branches at that node correspond to the possible values of the attribute, and each leaf node represents a classification or *decision* (Figure 3.18).

The basic idea is to pick an attribute K with values b_1, b_2, ..., b_s,, split the cases in the input casebase into subsets (classes) C_1, C_2, ..., C_s consisting of those cases that have the corresponding attribute value. Then if a subset has only cases in a single class (C_1, C_2, ... or C_s), that part of the tree stops with a leaf node labeled with that single class. If not, then the subset is split again, recursively, using a different attribute (Figure 3.18). In order to choose the best attribute to split on at any branch node, splitting criterion in ID3 (induction decision trees) named information gain theory is used.

The point of the ID3 algorithm (Quinlan, 1986) is to decide the best attribute, out of those not already used, on which to split the input cases that are classified to a particular branch node. The algorithm, in outline, is as follows:

1. If all the cases belong to a single class, a leaf node is created and labelled with the name of that class;
2. otherwise, for each attribute that has not already been used, the information gain that would be obtained by using that attribute on the particular set of cases classified to that branch is calculated.
3. Then the attribute with the greatest information gain is used as that branch node.

Splitting criterion requires the calculation of the information gain associated with using a particular attribute K. Suppose that there are r classes C_1, C_2, ..., C_r, and that of the N cases classified to this node, N_1 belong to class C_1, N_2 belong to class C_2, ..., and N_r belong to class C_r. If one example is selected at random from N and announced that the example belonged to class C_1. This announcement would have probability $p_1=N_1/N$, (and similarly $p_2 = N_2/N$, ..., and $p_r = N_r/N$) and the information it conveys is $-\log_2(N_1/N)$ bits (and similarly $-\log_2(p_2)$, ..., $-\log_2(p_r)$). As the probability goes up, the information conveyed goes to 0. It's highest for low probabilities. The term information represents the amount of information needed to identify the case as being a member of a particular class. Then the average amount of information needed to identify the class of a case in N is calculated as follows. This is called the entropy of N.

$$\text{Info (N)} = -\sum\left(\frac{N_r}{N}\times\log_2\frac{N_r}{N}\right) \tag{4.10}$$

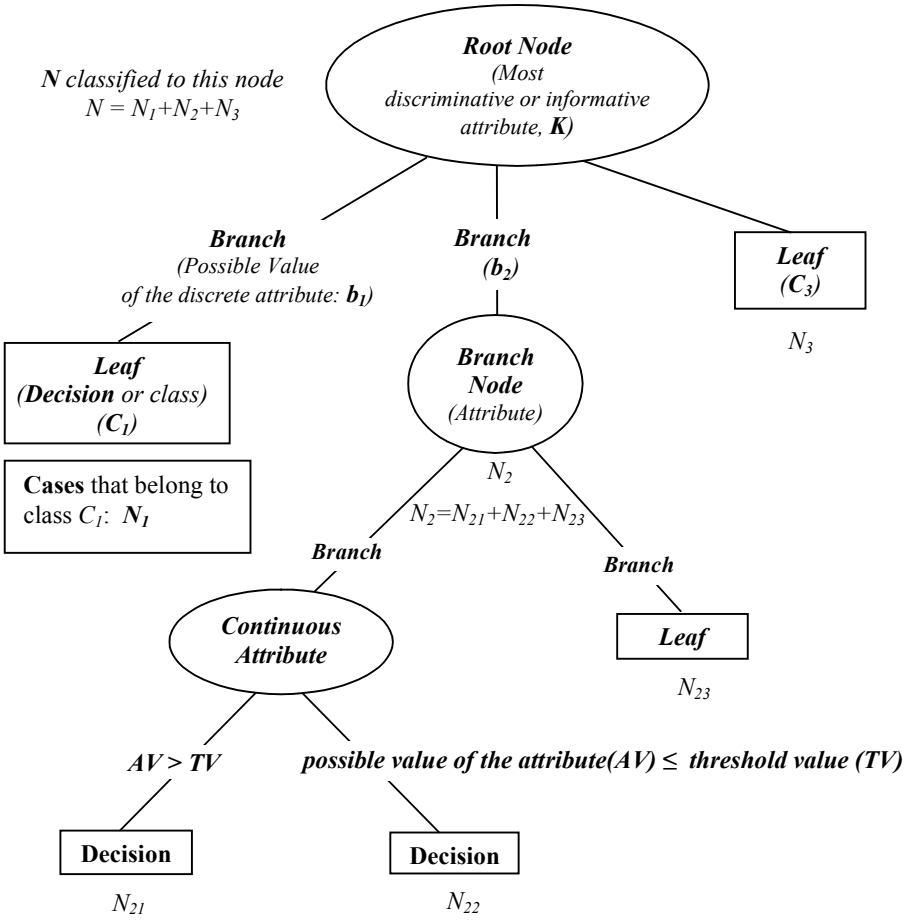


Figure 3.18. Basics of a decision tree

where the weighted sum is computed over the number of classes in N.

Suppose K attribute of the cases has b possible values. If K attribute were selected to be evaluated as a node in the tree, a decision tree node with b branches b_1, b_2, ..., b_s would be created. If N_{b_s} were the examples that have the value b_s for attribute K, the average entropy that resulted from making this split in the tree would be calculated as follows:

$$\text{Info}_K(N) = \sum \left(\frac{N_{b_s}}{N} \times \text{info}\left(N_{b_s}\right) \right) \tag{4.11}$$

where the sum is taken over the s possible values of the attribute K. This is computed for every attribute. Once the computations are done for every attribute, the attribute K that maximizes the value of info(N) – info_K(N) is selected. This difference (reduction in entropy) caused by portioning the cases according to attribute K is named the information gain for attribute K: InfoGain (N,K) = info (N) – info_K(N)

The process of handling the continuous valued attributes is slightly different. For continuous, rather than discrete attributes, the range is split into two groups: possible attribute values ≤ threshold and possible attribute values > threshold (Figure 3.18). The important issue is how to select the threshold. In order to determine the threshold, first the cases are sorted by the values of the attribute. Then the cases noting adjacent examples that belong to different classes are searched. The average values at those transition points are considered to be potential splits. Then each split found is evaluated by applying the information gain formula. The split that is best, which has the greatest information gain is selected. Accordingly a decision tree is constructed considering the information gain values of all kinds of attributes at hand.

See5 (1997) is a commercial software that is used for building decision trees. See5 builds a decision tree that consists of a sequence of logical decisions based on the attributes. It builds decision trees by employing a simple divide and conquer strategy as explained above. It first chooses an attribute as the current root, divides the input cases into subsets, and recursively tests the subsets, until all remaining cases belong to a single class. The choice of the attribute is based on the information gain. See5 always chooses the attribute with maximum information gain as the current root; such attributes

tend to be most discriminative or informative for classification at that point. The computations usually result in a small decision tree.

The three different decision tree learning algorithm methods used in this study are named (1) binary-dtree method, (2) info-top method, and (3) info-dtree method by Ling et al. (1997).

Binary-dtree method: Kibler and Aha (1987) first presented a simple approach that uses the presence or absence of attributes in the decision tree to determine their weights. If an attribute is present in the decision tree, then its weight is 1, otherwise its weight is 0. The method is very efficient since it only involves running See5 over the input cases. Cardie (1993) used this method to improve case-based learning and pointed out that a strategy of considering the positions of attributes in the decision tree (such as in the info-top and info-dtree methods) may work better.

Info-top method: Rather than considering only attributes with the maximum information gain (i.e., those appearing in the tree), this method considers the information gain of all attributes at the top level; that is, the information gain of all attributes based on all the input cases. Thus there is no need to construct the decision tree. These information-gain values are used as the weights in the similarity assessment process. Clearly, the attribute with maximum information gain is assigned a maximum weight, but other attributes can have some smaller effects in the similarity assessment as well, rather than being completely ignored.

Info-dtree method: This method takes into account the location of the attributes in the decision tree. Thus a decision tree is first constructed using the input cases. For each attribute, which may appear in several places in the tree, the weight is determined by taking the sum of its information gain at each appearance multiplied by the percent of input cases classified by that attribute. For example, if an attribute appears three times in the tree, with information gain values of 0.9, 0.8, and 1.0 with 40%, 20%, and 10% of the input cases classified by the attribute respectively, then the weight of this attribute is $(0.9 \times 0.4) + (0.8 \times 0.2) + (1.0 \times 0.1) = 0.62$. Clearly, attributes at lower levels contribute less to their weight because the number of input cases they classify is smaller. This method, like binary-dtree, considers only the information gain of those attributes that appear in the tree.

The attribute weights obtained are used in Step 4 of the CBR simulation process.

Step 4. *Calculating Weighted Case Similarities* – Case similarities are computed for each test case with respect to each input case by using the attribute similarities

calculated in Step 2 and the attribute weights generated in Step 3. For positive weights and normalized similarities, the weighted case similarities are always between 0 and 1, with a score of 1 indicating the case most similar to the test case and 0 the least. Weighted case similarities are computed according to the following formula:

$$CS_{ij} = \frac{\sum_{k=1}^{p} S_{ijk} \times w_k}{\sum_{k=1}^{p} |w_k|} \quad \text{for test case } i = (1, 2, ..., m) \text{ and input case } j = (1, 2, ..., n) \text{ for all}$$

$$\text{attributes } k = (1, 2, ..., p) \tag{4.12}$$

where CS_{ij}= Weighted case similarity between test case i and input case j over all the attributes k, S_{ijk} = Similarity between test case i and input case j for attribute k, and w_k = Weight of attribute k (See Figure 3. 19 for spreadsheet calculations).

Step 5*: Sorting Weighted Case Similarities and Corresponding Outputs* – The highest weighted case similarity CS_{ij} for a test case i indicates the closest matching input case j in the casebase. This operation is conducted (see Figure 3. 20) for each test case:

$$\max CS_{ij} = \max (CS_{i1}, CS_{i2}, ..., CS_{in}) \text{ for each i (i = 1, 2, … m)} \tag{4.13}$$

Once the highest weighted case similarities are identified for respective test cases (see Column AA in Figure 3.20 and 3.19), the corresponding case numbers and outputs are also listed (see Columns AB and AC in Figure 3. 20).

Step 6: *Calculating the Error* – The outputs listed in the preceding step (Column AC in Figure 3.20) are compared with the respective actual outputs (Column AD in Figure 3.20, same as Column H in Figure 3.13). The differences constitute the errors and are listed in Column AE in Figure 3.20. The average of the error values of all test cases is the overall error of the CBR process.

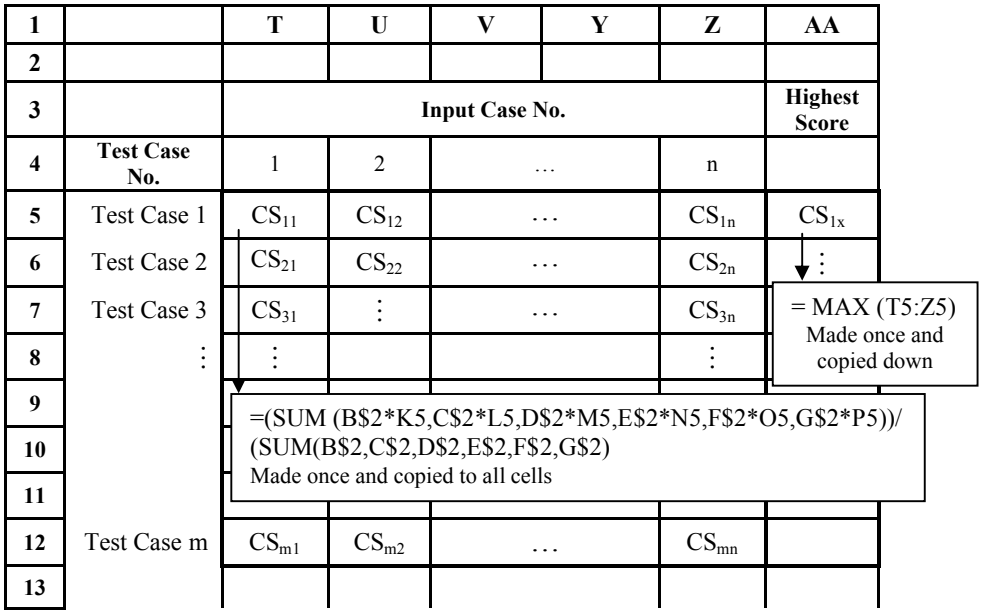


Figure 3.19. Case similarity matrix for all test cases

1		AA	AB	AC	AD	AE
2						
3		**Highest Score**	**Case No.**	**Output Value**	**Actual Outputs for Test Cases**	**Error**
4						
5	Test Case 1	CS_{1x}	x	O_x	O_1	E_{1x}
6	Test Case 2	CS_{2y}	y	O_y	O_2	E_{2y}
7	Test Case 3	⋮	⋮	⋮	⋮	
8	⋮					
9						
11	Test Case m	CS_{mz}	z	O_z	O_m	E_{mz}
12						$E_{average}$

=ABS((100-((AC5*100)/AD5))/100)

=AVERAGE(E_{1x} , E_{2y} , …, E_{mz})

Figure 3.20. CBR outputs and calculating the error

CHAPTER 4

FINDINGS AND DISCUSSION

ANN and CBR models and their integrated versions, which were developed in the previous chapter, are all tested by predicting the cost of the superstructure of residential building projects at an early design stage. Findings and analysis are presented in three sections. The first section analyzes the cost data used in this study. The second section includes the test results of the models of the case study and discussions of the findings. In the third section, a comparison of ANN and CBR Excel simulations are made.

4.1. Cost Data

Data used in this study belongs to a research report that investigated the cost of the structural system of 29 building construction projects undertaken in İstanbul, Turkey (Saner 1993). Analysis of the cost data revealed the main input parameters to be used in setting up the machine learning models. These parameters were the predominant cost drivers of the case (project) examples. The predominant cost drivers that could easily be identified in the early design stage were selected as the main parameters (Table 4.1) for modeling the machine learning techniques used in this study. They defined the buildings' formal characteristics and the amount of material required for the structural construction of the building. The total area bears a strong linear relation to the total cost of the building; and while it considerably impacts the structural cost, the ratio of the typical floor area to the total area of the building also becomes an important factor influencing directly the vertical section area of the load bearing frame. This in turn defines the cost of beams and columns. The number of floors is also clearly another important factor for the structural cost for its effect on the cost of columns. The ratio of the footprint area to the total area of the building is identified as one of the main key structural parameters, as it can be considered to be correlated with the width and depth of the foundation system. Foundations are classified as pier, wall or slab foundations to

Table 4.1. Main parameters (attributes) used in the prediction models

Input Attribute No	Attribute	Range
1	The total area of the building	330 m^2 – 3,484 m^2
2	The ratio of the typical floor area to the total area of the building	0.07 – 0.26
3	The ratio of the footprint area to the total area of the building	0.07 – 0.30
4	The number of floors	4 – 8
5	The type of overhang design	No overhang or one-way
6	The foundation system	Pier, wall, slab
7	The type of floor structure	Cast-in-situ concrete, precast concrete
8	The location of the core	At the sides, in the middle
Output	The cost of the structural system per square meter	\$30/m^2 – \$160/m^2

determine the effect of the volume of concrete and the amount of reinforcement on the total cost. The core of the building is composed of the vertical circulation system including stairs, elevators and the service duct. The examples in this case refer to two different locations of the building core: either in the middle or at the sides. To counteract the torsion effect, the structural system demands extra shear walls for the building cores located at the sides, which increases the total cost of the structural system. The buildings are analyzed to have either one-way or no overhang. The floor type of the apartments, whether reinforced concrete floor systems or precast concrete structural units is also considered to affect the structural cost.

Besides the variables considered above, there are some other important variables that have not been taken into account in the ML modeling. Since the selection of input variables significantly impacts the accuracy of the ML predictions, one may obtain different or better results if other possible important input variables are studied. The variables that could be investigated include the total height and the roof type of the building, quality classification of structural materials (concrete and steel), the ratio of the area of curtain walls to the total area of the vertical construction, the ratio of the number of secondary beams to the total number of beams in a typical storey of the building, etc. However only the variables that can easily be identified in the early design stage are considered in the current study.

4.2. Results and Discussion

ANN Results and Discussion

The ANN-Excel template was modified to suit the development of a cost model of residential building projects. With the inputs and outputs defined (see Figure 4.1), relevant data were entered for each project. The records of twenty-nine projects in Saner's (1993) study contain data on all the selected eight design variables and the corresponding cost of structural system per square meter.

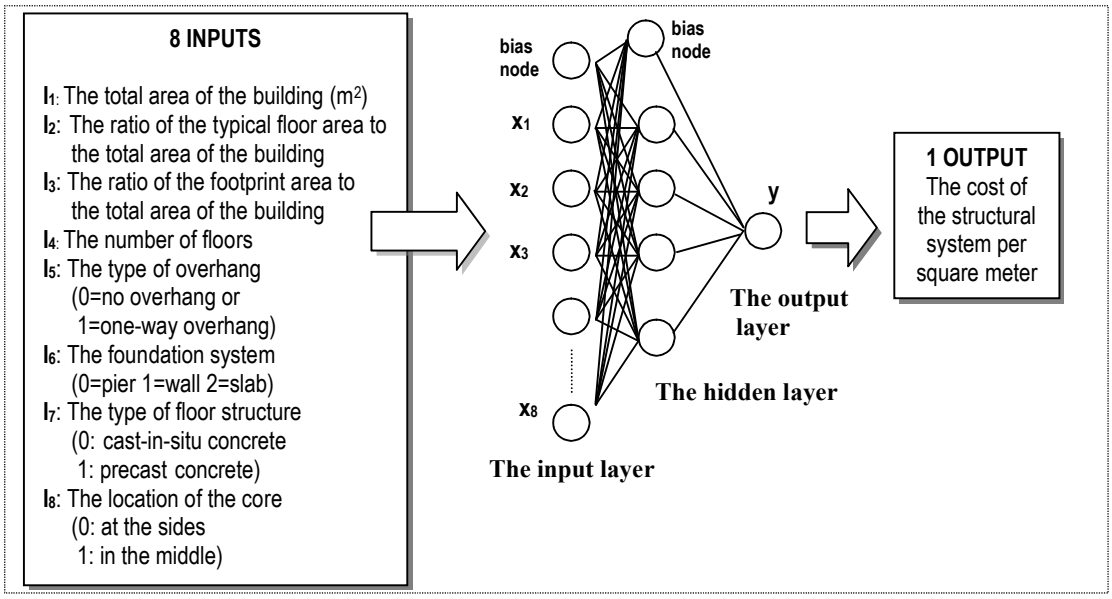


Figure 4.1. Description of ANN inputs and outputs

Using the described procedure for simulating ANN on Excel (in chapter 3), the data for 29 projects were then entered into Excel as shown in Figures 4.2, 4.3, 4.4, 4.5 and 4.6, with the qualitative values transformed into numbers according to the notations used in Figure 4.1. All ranges and matrices dimensions were modified according to the number of inputs, outputs, historical examples (past projects) and hidden nodes (i.e., N=8, O=1, P=21, and L=4, respectively). Using Solver and experimenting with various options on a trial and error basis, the resulting ANN weights were shown in Figure 4.4 and 4.5. Figure 4.6 shows the average error, 4.6%, obtained when Solver was used to optimize the model weights. By using GA and varying Evolver settings on a trial and error basis during the optimization process, Evolver was able to come up with an overall weighted error of 11%, with 0.5 weight on the training set and 0.5 weight on the testing set. For back-propagation training NeuroSolutions was used and several training experiments were conducted to arrive at the best-trained ANN. In these experiments network parameters such as the number of hidden layers, the number of hidden nodes, network connections, and the transfer functions were changed and the best results was documented (For model modifications, see Günaydın and Doğan 2004). After training, the ANN predictions were compared with the actual costs of the test cases. The minimum error when using NeuroSolutions was 7%.

The results of the ANN models using three different approaches for determining weights are presented in Table 4.2. The best overall model is the one produced by Excel Solver, providing excellent performance on both the training and test cases. While back-propagation training produced a network with small errors on the training cases, it behaved relatively poorly on the test cases. GA, on the other hand, did not produce good results probably because of its random selection of the generated population. Despite the consistent performance of the GA's model over the training and test cases, it exhibits a higher overall error. It is concluded, therefore, that the networks of simplex optimization and back-propagation training are most suited to the present case study.

CBR Results and Discussion

As an example application, the CBR-Excel template was populated by data collected from residential building construction projects. With the input attributes and the output attribute defined, relevant data were then entered into the CBR-Excel model

	A	B	C	D	E	F	G	H	I	J
1	**STEP 1: ORIGINAL UNSCALED INPUTS**									
	Project No.	**The total area of the building (m^2)**	**The ratio of the typical floor area to the total area of the building**	**The ratio of ground floor area to the total area of the building**	**The number of floors**	**The type of overhang design**	**The location of the core of the building**	**The floor type of the building**	**The foundation system of the building**	**The cost of the structural system per square meter**
3	1	675	0.2	0.182	6	1	0	1	1	49.87
4	2	1425	0.2	0.2	6	0	0	0	1	52.95
5	3	330	0.2	0.2	6	0	0	0	1	37.78
6	4	2025	0.14	0.13	5	0	1	1	1	62.12
7	5	1670	0.16	0.16	4	1	0	1	0	79.61
8	6	2082	0.16	0.3	6	1	0	1	1	58.72
9	7	3484	0.07	0.07	6	0	1	0	1	65.13
10	8	1364	0.25	0.23	6	0	1	0	0	76.36
11	9	1568	0.26	0.25	6	0	1	0	0	85.55
12	10	2533	0.16	0.16	6	0	1	0	2	51.04
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
23	21	569	0.16	0.14	6	1	0	0	1	42.49
24	22	1156	0.13	0.095	6	1	1	0	1	41.24
25	23	1146	0.202	0.19	5	0	0	0	2	127.7
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
31	29	2528	0.13	0.096	8	1	1	0	1	43.98
32	Min Value	330	0.07	0.07	4	0	0	0	0	35.47
33	Max Value	3484	0.26	0.3	8	1	1	1	2	151.9

Training Cases

Testing Cases

MIN(B3:B31)

MAX(B3:B31)

Figure 4.2. Step 1: Original unscaled inputs

	A	B	C	D	E	F	G	H	I	J
39	**STEP 2: SCALED INPUTS**									
40	**Project No.**	**The total area of the building (m^2)**	**The ratio of the typical floor area to the total area of the building**	**The ratio of ground floor area to the total area of the building**	**The number of floors**	**The type of overhang design**	**The location of the core of the building**	**The floor type of the building**	**The foundation system of the building**	**bias 1**
41	1	0.78123	0.368421	-0.026087	0	1	-1	1	0	1
42	2	-0.305644	0.368421	0.130434	0	-1	-1	-1	0	1
43	3	-1	0.368421	0.130434	0	-1	-1	-1	0	1
44	4	0.074825	-0.263158	-0.478261	-0.5	-1	1	1	0	1
45	5	-0.150285	-0.052632	-0.217391	-1	1	-1	1	-1	1
46	6	0.110970	-0.052632	1	0	1	-1	1	0	1
47	7	1	-1	-1	0	-1	1	-1	0	1
48	8	-0.344325	0.894736	0.391304	0	-1	1	-1	-1	1
49	9	-0.214965	1	0.565217	0	-1	1	-1	-1	1
50	10	0.396956	-0.052632	-0.217391	0	-1	1	-1	1	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
61	21	-0.848446	-0.052632	-0.391304	0	1	-1	-1	0	1
62	22	-0.476221	-0.368421	-0.782609	0	1	1	-1	0	1
63	23	-0.482562	0.389473	0.043478	-0.5	-1	-1	-1	1	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
69	29	0.393785	-0.368421	-0.773913	1	1	1	-1	0	1

2*(B3-B$32)/(B$33-B$32)-1

Figure 4.3. Step 2: Scaled inputs

74	STEP 3: WEIGHTS OF LINKS FROM 8 INPUTS AND A BIAS TO 5 HIDDEN NEURONS									
75	1	1.029	1.000	1.0310	0.998	1.001	-5.866	1.113	0.993	0.921
76	2	0.830	2.685	0.5498	-0.78	1.338	0.712	-0.721	0.515	-0.43
77	3	0.206	1.501	0.9296	0.035	-1.529	0.133	0.134	3.240	0.783
78	4	-0.847	3.662	4.6076	1.573	3.119	3.497	-0.239	-1.162	2.918
79	5	0.1200	2.7294	2.9047	0.281	3.460	1.255	-0.770	0.838	1.727

Figure 4.4. Step 3: Weight of links from 8 inputs to 5 hidden neurons

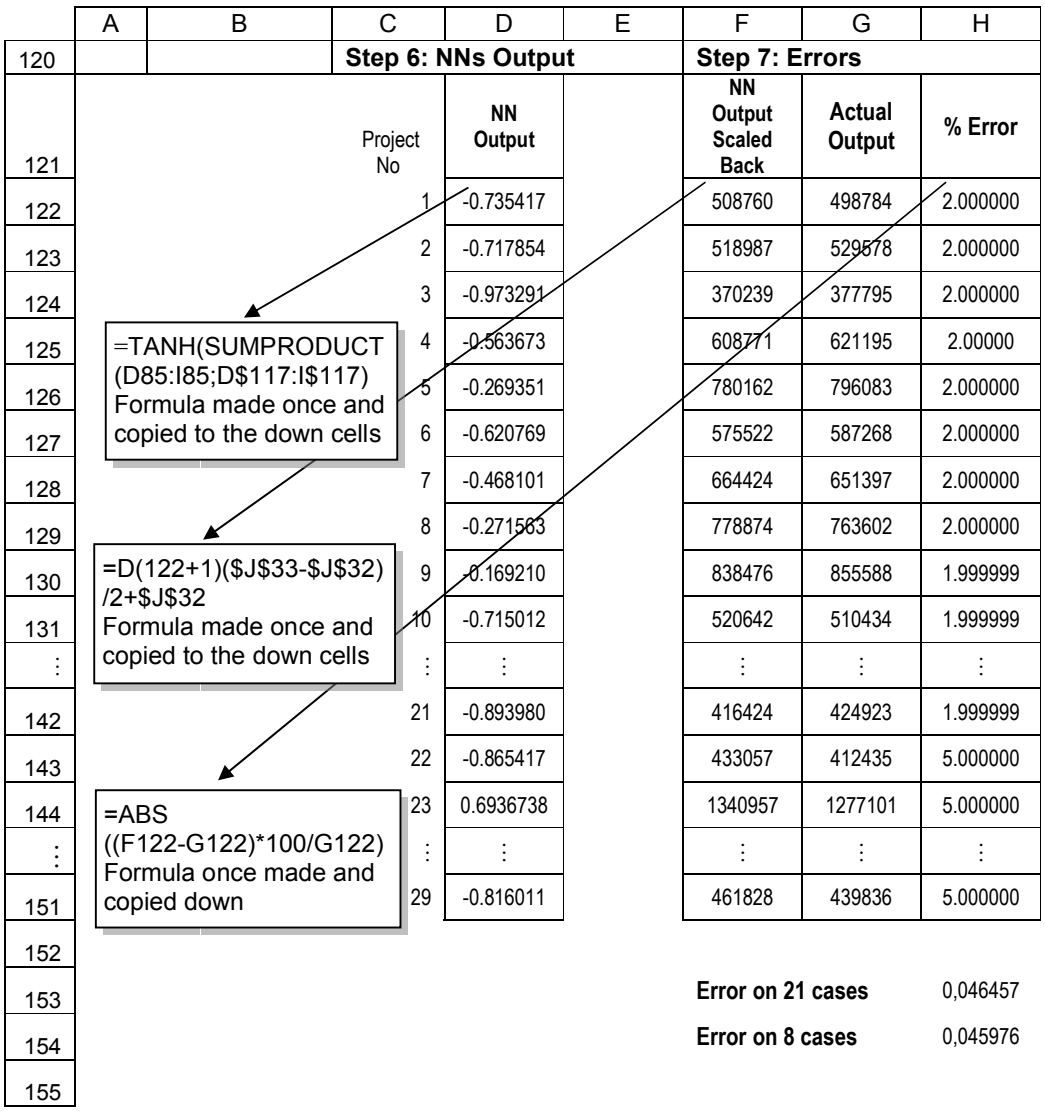
	A	B	C	D	E	F	G	H	I
83			**STEP 4: Outputs of hidden neurons**						
84			Project No	1	2	3	4	5	Bias 2
85			1	0.999999	-0.204478	-0.36117	0.9995430	0.9993270	1
86			2	0.999880	-0.744214	0.990207	-0.848484	-0.701478	1
87			3	0.999500	-0.911489	0.986969	-0.580060	-0.741386	1
88			4	-0.999988	-0.980030	0.940175	-0.744563	-0.998134	1
89			5	0.999995	-0.569430	-0.99965	0.6764327	0.8479233	1
90			6	1	-0.033210	0.127449	0.9999964	0.9999861	1
91			7	-0.999999	-0.991813	0.088833	-0.999971	-0.999941	1
92			8	-0.999998	0.900371	0.609595	0.9999999	0.9949690	1
93			9	-0.999996	0.961079	0.783645	1	0.9989984	1
94			10	-0.999986	0.237973	0.999955	0.6872906	0.3832194	1
⋮			⋮	⋮	⋮	⋮	⋮	⋮	⋮
105			21	0.999954	-0.150388	-0.92635	0.9056252	0.9973980	1
106			22	-0.999997	0.477334	-0.97192	0.9999418	0.9991446	1
107			23	0.999926	-0.185502	0.999981	-0.997652	-0.370278	1
⋮			⋮	⋮	⋮	⋮	⋮	⋮	⋮
113			29	-0.999869	0.428230	-0.95645	0.9999899	0.9996242	1

=TANH (SUMPRODUCT (B41:J41,B$75:J$75)) Formula once made and copied down to cells

=TANH(SUM PRODUCT (B41:J41,B$79:J$79)) Formula once made and copied down to cells

116	**STEP 5:** Weights from 5 hidden neurons to 1 output neuron						
117	1	-1.259155	-2.92964	1.4595452	-5.296362	7.8224082	-2.2764

Figure 4.5. Step 4: Outputs of hidden neurons and
Step 5: Weights from 5 hidden neurons to 1 output neuron



Project No	NN Output	NN Output Scaled Back	Actual Output	% Error
1	-0.735417	508760	498784	2.000000
2	-0.717854	518987	529578	2.000000
3	-0.973291	370239	377795	2.000000
4	-0.563673	608771	621195	2.00000
5	-0.269351	780162	796083	2.000000
6	-0.620769	575522	587268	2.000000
7	-0.468101	664424	651397	2.000000
8	-0.271563	778874	763602	2.000000
9	-0.169210	838476	855588	1.999999
10	-0.715012	520642	510434	1.999999
⋮	⋮	⋮	⋮	⋮
21	-0.893980	416424	424923	1.999999
22	-0.865417	433057	412435	5.000000
23	0.6936738	1340957	1277101	5.000000
⋮	⋮	⋮	⋮	⋮
29	-0.816011	461828	439836	5.000000

Figure 4.6. Step 6: NN outputs and Step 7: Errors

Methods for determining ANN weights	Average error in ANN prediction
Simplex optimization	4%
Back-propagation (gradient descent)	7%
Genetic Algorithms (GA)	11%

Table 4.2. Average error percentages for ANN models

using the procedure described in Figure 3.13, 3.14, 3.18 and 3.19. A set of test cases were used to evaluate the effect of the attribute weights generated by all six methods (Figure 3.15 3.16, 3.17, 4.11), the dataset of 29 projects being randomly split into an *input* set containing 24 projects, and a *test* set containing 5 projects. In other words, CBR-Excel simulation described in chapter 3 was modified as follows: there were $i = 1, 2, ..., (m = 5)$ projects that were used as test cases, $j = 1, 2, ..., (n = 24)$ projects as input cases, $k = 1, 2, ..., (p = 8)$ input attributes, and one output. The impact of the six sets of attribute weights was evaluated using the same test set of 5 projects.

The next step was to set up the Excel template to calculate attribute weights. The overall error obtained in CBR is a function of attribute weights. The attribute weights were generated by using (1) the feature counting method, (2) the gradient descent method (3) genetic algorithms, (4) binary-dtree method, (5) info-top method, (6)info-dtree method. The attribute weights obtained (Table 4.3) were input into the CBR-Excel application.

For the methods using decision tree learning algorithms, See5 was used to generate attribute weights for the CBR model. The decision tree constructed by See5 is presented in Figure 4.11. Each *branch* node (oval shape) in the decision tree represents an attribute, and the branches correspond to the possible values of the attribute. Each *leaf* node (rectangular shape) represents a decision or a class (Table 4.4). The attribute weights obtained by using decision trees (Table 4.3) were put into CBR-Excel application (Figure 4.7, 4.8, 4.9 and 4.10) similar to the other sets of weights generated by other methods mentioned.

After the attribute weights were determined by using feature counting, gradient descent, GA, and ID3 (decision tree) learning algorithm methods (binary-dtree, info-top and info-dtree), the CBR-Excel model was run and the performance of the model was evaluated vis-à-vis each method. The results presented in Table 4.3 indicate that the GA-augmented CBR model yielded an average error of 16.23% whereas feature counting+CBR and info-top+CBR yielded an average error of 17.63% and binary- dtree

1	A	B	C	D	E	F	G	H	I	J
2	Weights	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	
3	TEST CASEBASE									
4	**Test Case No.**	**Input Attributes**								**Output Attribute**
5		1	2	3	4	5	6	7	8	
6	1	2969	0.14	0.120	7	no cons	middle	RC	slab	109.35
7	2	1238	0.16	0.160	5	no cons	sides	RC	slab	37.66
8	3	2082	0.16	0.300	6	one-way	sides	pre-cast	wall	58.72
9	4	2528	0.13	0.096	8	one-way	middle	RC	wall	43.98
10	5	1172	0.16	0.160	4	no cons	middle	RC	slab	74.84
11										
12	INPUT CASEBASE									
13	**Input Case No.**	**Input Attributes**								**Output Attribute**
14		1	2	3	4	5	6	7	8	
15	1	675	0.20	0.182	6	one-way	sides	pre-cast	wall	49.87
16	2	2861	0.16	0.080	7	no cons	middle	RC	slab	62.70
17	3	330	0.20	0.200	6	no cons	sides	RC	wall	37.77
18	4	1425	0.20	0.200	6	no cons	sides	RC	wall	52.95
19	5	964	0.17	0.150	5	one-way	middle	RC	slab	103.04
20	6	1314	0.15	0.140	6	no cons	sides	RC	wall	37.97
21	7	3484	0.07	0.070	6	no cons	middle	RC	wall	65.13
22	8	1364	0.25	0.230	6	no cons	sides	RC	pier	76.36
23	9	1568	0.26	0.250	6	no cons	middle	RC	slab	85.55
24	10	2533	0.16	0.160	6	no cons	middle	RC	slab	51.04
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
38	24	1518	0.13	0.120	7	no cons	sides	RC	wall	36.38
39										

Figure 4.7. Formatting data to a case spreadsheet

=MIN(B6,B$15)/MAX(B6,B$15)
Made once and copied for all cells with numerical information

=IF(F6=F$15,"1","0")
Made once and copied for all cells with textual information

1	K	L	M	N	O	P	Q	R	S
2									
3	Input Case No.	Input Attributes							
4		1	2	3	4	5	6	7	8
5	1	0.227	0.700	0.659	0.857	0	0	0	0
6	2	0.964	0.875	0.667	1.000	1	1	1	1
7	3	0.111	0.700	0.600	0.857	1	0	1	0
8	4	0.480	0.700	0.600	0.857	1	0	1	0
9	5	0.325	0.824	0.800	0.714	0	1	1	1
10	6	0.443	0.933	0.857	0.857	1	0	1	0
11	7	0.852	0.500	0.583	0.857	1	1	1	0
12	8	0.459	0.560	0.522	0.857	1	0	1	0
13	9	0.528	0.538	0.480	0.857	1	1	1	1
14	10	0.853	0.875	0.750	0.857	1	1	1	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
28	24	0.511	0.929	1.000	1.000	1	0	1	0
29									

Figure 4.8. Attribute similarity matrix for Test Case 1
(5 similar matrices are generated, one for each of the 5 test cases)

=(SUM(B$2*L5,C$2*M5,D$2*N5,E$2*O5,F$2*P5,G$2*Q5,H$2*R5,I$2*S5))/(SUM(B$2,C$2,D$2,E$2,F$2,G$2,H$2,I$2)
Made once and copied to all cells of the matrix

= MAX (U5:AF5)
Made once and copied down

1	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG
2														
3	Test Case No.	Input Case No.												Highest Score
4		1	2	3	4	5	6	7	8	9	10	…	24	
5	1	0.283	0.939	0.418	0.470	0.724	0.525	0.637	0.440	0.799	0.920	⋯	0.566	0.939
6	2	0.505	0.691	0.616	0.700	0.706	0.737	0.398	0.673	0.722	0.771	⋯	0.676	0.961
7	3	0.833	0.343	0.658	0.731	0.431	0.717	0.481	0.503	0.381	0.403	⋯	0.693	0.833
8	4	0.579	0.605	0.499	0.560	0.587	0.606	0.740	0.317	0.463	0.581	⋯	0.660	0.898
9	5	0.352	0.815	0.460	0.536	0.834	0.572	0.520	0.509	0.840	0.892	⋯	0.515	0.984
10														

Figure 4.9. Case similarity matrix for all test cases

=ABS((100-((AK4*100)/AL4))/100)
Made once and copied down

1	AH	AI	AJ	AK	AL	AM
2						
3	Test Case No.	Highest Score	Input Case No.	Output Value	Actual Outputs for Test Cases	Error
4	1	0.939	18	62.70	109.35	0.43
5	2	0.961	21	41.58	37.66	0.10
6	3	0.833	1	49.87	58.72	0.15
7	4	0.898	11	41.24	43.98	0.06
8	5	0.984	2	64.50	74.84	0.14
9						0.176

=AVERAGE(AM4:AM8)

Figure 4.10. CBR outputs and error

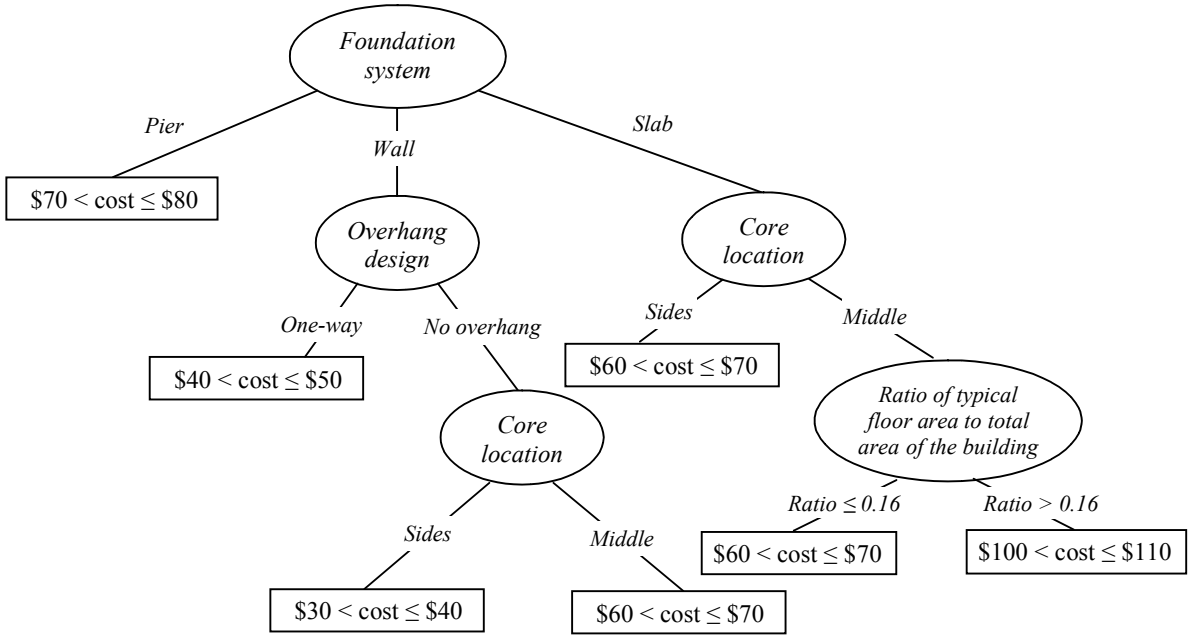


Figure 4.11. Decision tree constructed by See5 according to the output attribute classes in Table 4.4.

Table 4.3. Optimized attribute weights for CBR Excel model and average error percentages

Weight Generation Method	Attribute Weights								Average error in CBR prediction
	Total area	Ratio of floor area to total area	Ratio of footprint area to total area	Number of floors	Overhang design	Core location	Floor type	Foundation system	
Feature Counting	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	17.63%
Gradient Descent	0.0069	0.1885	0.1528	0.1427	0.1049	0.1560	0.0316	0.2161	21.20%
Genetic Algorithms	1.0000	2.0056	1.0010	9.9988	1.0031	1.0000	3.9999	1.0000	16.23%

Table 4.3. (continued) Optimized attribute weights for CBR Excel model and average error percentages

Decision tree method of weight generation	Attribute Weights								Average error in CBR prediction
	Total area	Ratio of floor area to total area	Ratio of footprint area to total area	Number of floors	Overhang design	Core location	Floor type	Foundation system	
Binary-dtree	0	1	0	0	1	1	0	1	20.70%
Info-top	0,387129	0,451902	0,439009	0,355676	0,509398	0,511249	0,189805	0,783560	17.63%
Info-dtree	0	0,204025	0	0	0,243221	0,604721	0	0,783560	20.70%

Table 4.4. Classes specified for output attribute of cost per square meter

Class No	Cost
1	$\$30/m^2 < Cost \leq \$40/m^2$
2	$\$40/m^2 < Cost \leq \$50/m^2$
3	$\$50/m^2 < Cost \leq \$60/m^2$
4	$\$60/m^2 < Cost \leq \$70/m^2$
5	$\$70/m^2 < Cost \leq \$80/m^2$
6	$\$80/m^2 < Cost \leq \$90/m^2$
7	$\$90/m^2 < Cost \leq 100/m^2$
8	$\$100/m^2 < Cost \leq \$110/m^2$
9	$\$110/m^2 < Cost \leq \$120/m^2$
10	$\$120/m^2 < Cost \leq \$130/m^2$
11	$\$130/m^2 < Cost \leq \$140/m^2$
12	$\$140/m^2 < Cost \leq \$150/m^2$
13	$\$150/m^2 < Cost \leq \$160/m^2$

+CBR and info-dtree+CBR had average errors of 20.70%; and the gradient descent+CBR had average error of 21.20%.

The setting up of the attribute weights in the **feature counting** method was straight forward in that all weights were taken as 1. In the **gradient descent** method, the experimentation between the arithmetic and geometric decrementation approach showed that the geometric approach resulted in better predictions. Default parameters were used for all other factors following the recommendations of the software developer (Esteem 1994).

GA optimization could have been performed with multiple "evaluation cases." But the selection of five test cases out of a total of 29 limited the number of cases in the input casebase to as few as 24, which in turn necessitated the selection of very few "evaluation cases" (see Figure 3.16), in this study only one. Finally, in the GA optimization process, the weight of each attribute was constrained between 1 and 10. Using a range of 0 to 10 rather than 1 to 10 could have had the effect of eliminating certain attributes, hence making the process more efficient. It is worth exploring this issue in future research.

After the first cycle of the GA optimization process, the "evaluation case" was returned to the input casebase and the next case picked for the next cycle of GA optimization. In other words, every input case in the input casebase was used once as an "evaluation case." Since the similarity of two identical cases is indicated by 1 in the CBR system, the objective function of the GA optimization was set to make the case similarities closer to 1.

Three different approaches were experimented in the GA optimization process to improve prediction accuracy. In the first approach, the objective of GA optimization at every cycle was to maximize the weighted case similarity of the input case that had the highest similarity with the "evaluation case." In the second approach, the objective was to maximize the average weighted case similarity of all the 23 cases that were considered at each cycle. The third approach involved maximizing the weighted case similarity of the input case whose output was closest to the output of the "evaluation case." The GA optimization process was performed for 24 cycles, each using these three approaches. The averages of the attribute weights determined in these 24 cycles were used in the CBR prediction model. The optimized attribute weights were calculated as follows:

$$w_k = \frac{\sum_{j=1}^{n} w_{jk}}{n} \tag{4.1}$$

where k = 1, 2, … , 8 attributes and n = 24 input cases

The setting up of the attribute weights in the **binary-dtree method** was straight forward in that the attributes appearing in the decision tree (the foundation system, the type of overhang design, the location of the core, and the ratio of the typical floor area to the total area of the building) (see Figure 4.11) were weighted as 1, whereas the attributes that did not appear in the tree (the total area of the building, the ratio of the footprint area to the total area of the building, the number of floors, the type of floor structure) were weighted as 0, as seen in Table 4.3. In the **info-top method**, all of the 8 attributes were given weights according to their information gain values. The attribute with the highest information gain value is selected as the root of the decision tree by See5. In this study, the foundation system with the information gain value of 0,783560 was selected as the root (see Figure 4.11). The information gain values of all the attributes (i.e., their weights) are presented in Table 4.3. In the **info-dtree method**, the attributes that appear in the tree constructed by See5 (Figure 4.11) were given weights in consideration of their information gain values and their positions in the tree. For example, the attribute "console direction" appeared twice in the tree, with information gain values of 0.750 and 0.918 with 50% and 25% of input cases classified by the attribute respectively; the weight of this attribute is calculated as (0.750 × 0.5) + (0.918 × 0.25) = 0.6045. The weights of the attributes in the decision tree were calculated using the same principle and are presented in Table 4.3.

As discussed by Ling et al. (1997), if the number of input cases is small, See5 constructs an overly simple decision tree, overlooking relevant attributes. In the case study presented in this paper, there were 3 continuous and 5 discrete attributes but only 29 cases. Because 5 cases had to be used as test cases, only 24 cases were left as input cases. As a result, See5 constructed a tree that included only four attributes. When this happens, the performance of the binary-dtree and info-dtree methods (which consider only the attributes in the decision tree) is bound to be worse than the info-top method (which considers the information gain of all attributes). It was therefore not surprising to find out that binary-dtree + CBR and info-dtree + CBR did not generate predictions

that are as strong as the prediction generated by the info-top + CBR alternative because they only use the attributes that appear in the decision tree and therefore do not take into account the information gain of the other relevant attributes even though it is likely that such information gain affects the classification of some cases used in the study. Our findings support the conclusion of Ling et al. (1997) that info-dtree and binary-dtree are immune to irrelevant attributes, and that info-top is suitable for situations where there are not enough input cases and where all attributes may be relevant.

On the other hand, it was surprising to see that binary-dtree + CBR performed as good as info-dtree + CBR. After all, info-dtree is considered to be a more sophisticated method than binary-dtree that assigns a weight of 1 to all attributes in the decision tree, regardless of their position in the tree (Ling et al. 1997). While binary-dtree was found to be as effective as info-dtree in this case study, it must be noted that a limited number of input cases were available. The performance of the info-dtree method could possibly improve with larger numbers of input cases.

The performance of the optimized attribute weights were tested on the five test cases. Out of the six approaches, GA approach performed best. The attribute weights presented in the last row of Table 4.3 were obtained by using the GA approach.

One of the reasons why the average errors obtained (last column in Table 3) were not very low had to do with the nature of the output attribute. The output of the cases considered in this study was the unit cost of construction of the superstructure and its value ranged between \$30 and \$160/m^2 (see Table 4.1 and Table 4.4). In order to have high prediction accuracy, one should have at least two or more cases with not only quite similar input attributes but also almost identical outputs, which is most improbable given the small number of cases (total 29) that were available for this study and the wide range of unit costs associated with the cases considered. The average errors reported in this study could have been lower had the output variables been binary or had there been a larger number of cases with an output attribute whose value varied in a smaller range.

4.3. Comparison of ANN and CBR Excel Simulations

This study has evaluated ML techniques of ANN and CBR and their integrated (augmented) forms, which were used to make cost estimations. These have been

compared with their prediction accuracy. However, there are other characteristics of these techniques that will have an equal, if not greater, impact upon their adoption. Below the relative merits and demerits of those are discussed. In light of the studies conducted with both ANN simulation and CBR simulation, five factors are considered to assess their utility: preprocessing effort, configurability, explanatory value, accuracy and improvement potentials.

4.3.1. Preprocessing Effort for Conversion of Data

Data consist of cases and their related features. The content could be both in numerical and textual values. The techniques of handling data for ANN and CBR systems are different. The ANN system can only handle numerical values, which also need to be scaled to a certain range. Conversions of numerical and textual input data are essential to suit ANN processing. (The numerical values are often scaled to a range from [-1,1] for tanh activation function to avoid fluctuations in the mathematical calculations of an ANN system.) Although both CBR and ANN systems require the organization of data into a matrix form to suit the Excel format, the ANN system needs 3 more steps in this procedure in order to be able to process input data and produce meaningful output data. This certainly brings additional preprocessing effort for organization of data. The spreadsheet simulations bring the advantage of transparency, however they cannot avoid the considerable time required to build them up, when compared with commercial software. Therefore, the ANN system may be less advantageous when dealing with a large data set. Then it is easier to use CBR which handles cases in their original representations, without converting from one type to another. This may also be important in order to prevent loss of information since learning performance may deteriorate when modified data are used (Reich 1997). In this study, the building cost data were in the form of both numerical and textual values. Features expressed as text were used in the CBR study. Textual data were subjected to numerical transformation in a continuous manner in the ANN study; the numerical data were reduced to a range [-1,1] with a linear scaling formula.

4.3.2. Configurability in Spreadsheet Format

The second major factor in comparing ANN and CBR prediction systems for this simulation study is configurability, in other words how much effort is required to build the prediction system in order to generate useful results. Considering the preprocessing effort mentioned previously for conversion of data, CBR needs relatively little effort. However, model building is a more complex issue than entering and converting data. The ANN model needs specifying the number of hidden layers, hidden neurons, bias nodes, a learning algorithm and a transfer function for the Excel format, whereas CBR only needs specifying the feature and case similarity functions. These variables are the tools of modeling, which analyst uses to find the optimum combinations and results. Although various sets of books have been published on ANN modeling, the process is agreed to be largely one of trial and error. Therefore, it is obvious that it takes considerable effort to configure the neural network architecture and it certainly requires a fair degree of expertise. For this reason, it is difficult to see how an ANN model could easily be built up within the spreadsheet format by analysts, where the analyst has to manually enter all the values, build up the model, evaluate the performance and then accordingly rebuild the model again and again until he/she gets an optimum solution. This is generally related with the burden of the training process of an ANN system. However, the burden could be intolerable in a spreadsheet format. CBR, on the other hand, does not require the combinations of parameters to build up its prediction model. Since it does not predict from scratch, but retrieves cases from a case-base, it uses simple feature similarity and case similarity formulas, which can be made once in Excel and easily copied to all cells thereafter.

4.3.3. Accuracy for Cost Prediction

Not generating data from scratch but adjusting from a case-base enhances the configurability of CBR in Excel format, but it appears to be a disadvantage in this particular study since there are only few examples to store in the case-base. Consequently in this study, the ANN model was able to produce closer cost values to actual costs than the CBR model (see Table 4.2 and Table 4.3). It was obvious that even though, CBR had worked with full efficiency and selected the closest cost value, it

definitely would never have been able to predict better than what existed as the closest cost in its case-base. Although several methods by utilizing highest score ranks were applied in order to get closer predictions, none produced better results. If the neural network paradigm is suitable for the data available, a key aspect of many ANN models is that they are able to learn, and their behavior may improve with training and experience (Barrow 1996). In this case this advantage of ANN provided superior prediction results over CBR.

4.3.4. Explanatory Value

Although ANN models are great learners, almost like humans, the rules behind their judgment is not explainable. One attraction of the transparent spreadsheet simulations carried out in this study is that the analyst is able to see and control all the formulas and connections being used by the prediction system. However, in an ANN system if a particular prediction is in some sense surprising to the analyst, it is harder to establish any rationale for the value generated. It is difficult to evaluate the outcome of an ANN study merely by studying the network architecture and neuron weights. By comparison, CBR appears to offer an advantage in this respect. Unlike reducing error by weight generation through back-propagation learning in ANN, CBR estimates by analogy. Cases are ordered in degree of similarity to the target case by utilizing similarity assessment methods calculated by assigning weights to the related features. Indeed, above the explanatory value, this technique encourages the participation of the analyst for getting better and effective predictions.

4.3.5. Improvement Potential via Integration of Other Methods

For better and effective predictions, weights are the important adjustable variables that can be freely manipulated on an Excel spreadsheet. Both in ANN and CBR, the weights of the variables are the adjusted in order to build up the optimum prediction system. Therefore, the improvement potential of these models are strongly tied to how realistic the weight of the variables are. In the studies of ANN and CBR, cited in Chapter 3, the optimization of the weights is done by well established methods (described in Chapter 3).

When comparing the model building effort for the two systems, it was mentioned that the primary advantage of CBR over ANN was that a CBR application did not need to be trained (Kasravi 1994). On the other hand, in the GA/CBR study, the selection of the weights for the similarity assessment method turned not to be an important operation, which consumed as much time as the training procedure of the ANN model. By comparison, GA integration in ANN is a simpler procedure, which is carried out only once for the whole training cycle. On the other hand, weight generation in CBR is a critical issue on which the success of the CBR technique heavily relies. The GA optimization for feature weights in this CBR study was carried out once for each case in the case base in order to get the most benefit out of their integration.

For the study carried out with GA/ANN, the GA optimization of weights was not more successful than the simplex optimization method or back-propagation training (Günaydın and Doğan 2004, Doğan et al. 2005c). However, GA offered several improvements in the GA/CBR study. GA was able to reduce the effect of less important features; and it was able to eliminate the unimportant features when constraints were scored on a scale starting with 0. This means that if a feature is of no importance, it was assigned a 0 weight by GA. In the study carried out by Doğan et al. (2005a) it was found out that every feature could somewhat improve the prediction accuracy, so the constraints were set to begin from 1. Irrelevant features for ANN models are also an important problem investigated lately by Shi (2004).

The feature counting method which assigns weight values of 1 to all attributes required no effort on the part of this researcher. But all the other methods (decision tree learning algorithms, gradient descent and GA) required a far greater effort to generate the optimized weights that were later plugged into the Excel model. The ANN model was more welcoming than CBR when the Excel add-in programs were used to determine ANN weights.

Whatever mechanism is being utilized, it is clear that although accuracy is the most important concern, it is not sufficient to consider the accuracy of prediction systems in isolation. The consistency (explanatory value), continuity (configurability and preprocessing effort) and improvement of the systems are also of great importance.

4.3.6. Conclusions

CBR and ANN models were used by Doğan et al. (2005a, 2005b); and Günaydın and Doğan (2004) and Doğan et al. (2005c), respectively, to predict the early cost estimate of residential building projects. A comparison of the experiences with the development of CBR and ANN models shows the following:

- The case study used in the models to compare ANN and CBR indicates that augmented ANN and CBR models by different weight generation methods may make better predictions than standard methods provided by commercial software of ANN or CBR (Doğan et al. 2005a, 2005b, 2005c). However, in both cases, the model building process is unnecessarily cumbersome for Excel simulations. Even after the systems are designed, when they need to be updated with new cases for the long-term use of these models, it is even more cumbersome since all the model building process should be repeated and tested with each update. This is the reason that the automation is importantly needed. Currently, there is no commercial software like GA-CBR. However Jarmulak et al. (2000) reported working such integration on the CBR software ReCall (1993). For the ANN system, some software is supported by genetic training, e.g., NeuroShell (2002). Augmenting CBR weights with different decision tree learning algorithm methods is discussed in some articles (Ling et.al. 1997) published in the computer science field; but there is no software designed for their integration as yet. The CBR software Esteem only supports a limited part of this kind of integration just by considering the numerical attributes. When input attributes of data include textual information, Esteem is unable to take those attributes into consideration when performing its prediction.

- Even after the release of integrated software, more research should be carried out for different data sets because specific recommendations are needed as to which approach could be more appropriate in what type of domain [for what type of output (numeric, textual, binary, etc...)] or for

what type of input data (i.e. number of inputs /attributes and training and retrieving case numbers). This type of guideline would be of great help to the developers of prediction models.

- The early stage cost estimation effort conducted by using different machine learning applications has a number of distinct characteristics compared to other prediction problems. First, the training set is comparatively small. Second, the predictions generally have a higher degree of significance to the analyst. This has the consequence that interaction or collaboration, between the prediction system and the analyst is of great importance. Allowing the analyst to participate in the prediction process by utilizing spreadsheet simulations may lead to two beneficial effects. First, it may enhance accuracy. Analysts may provide some kind of sanity check on the systems, while the system allows them to manipulate far more characteristics manually than would be possible by commercial software. Second, it may increase confidence in the prediction. This consideration is also important in order to avoid the situation where end-users reject a prediction system.

In this dissertation two machine learning techniques augmented with various weight generation methods for predicting early cost estimation of superstructure of buildings were compared. These techniques were compared in terms of preprocessing effort, accuracy, explanatory value, configurability and improvement potentials. Despite finding that there are differences in prediction accuracy levels, it is argued that it may be the other characteristics of these techniques that may have an impact upon their adoption. It was found that the explanatory value of estimation by analogy gives CBR an advantage when considering its interaction with the analyst and end-users. It was also found that problems of configuring neural networks tend to rather counteract their superior performance in terms of accuracy. This preliminary research has shown that the machine learning (ML) techniques used in this study are locally significant but are not generalizable. It is believed that it is important to further investigate these ML methods, particularly to explore under which conditions they are most likely to be effective.

CHAPTER 5

CONCLUSIONS

This dissertation has presented the developments and findings of ANN and CBR models for the prediction of unit structural cost of residential buildings. For doing so, the basics of artificial neural networks and cased reasoning processing are analyzed in the context of cost prediction. An ANN spreadsheet model has been developed based on Hegazy and Ayed's (1998) Excel template. A CBR Excel model has been developed following the spreadsheet based user interface of a commercial CBR software (Induce-It, 2000). Cost data belonging to residential buildings in İstanbul have been used to test the models. An investigation of the impacts of weight generation methods on the ANN and CBR models in the building cost prediction domain has been conducted. Various methods including simplex optimization, back propagation training, and genetic algorithms for ANN and feature counting, gradient descent, genetic algorithms (GA), binary-dtree, info-top and info-dtree for CBR model have been used. Thus, the two main Excel models of ANN and CBR developed in this study produced nine different models. Spreadsheet structures of the developed ANN and CBR models made them flexible for weight generation alterations and further development.

This research provides contributions in several areas. The following paragraphs itemize conclusions and major identifiable tasks that have been accomplished. At a global level, this dissertation developed a unique methodology by using machine learning (ML) methods for improving cost prediction at the early design stage of building construction. The following are the research findings, the conclusions and contributions:

1. The review and the results of the study show that cost prediction at the early design stage can be enhanced by major breakthrough developments in machine learning (ML) domain. Architects and project managers involved in the process of building design and construction may take advantage of ML techniques for higher cost prediction accuracy and therefore for higher quality in building design and construction processes.

2. The study has been able to introduce alternative approaches of using ANN and CBR models for higher cost prediction of the structural system at the early stages of the building design process. Both of the approaches have been capable of providing high prediction accuracy (96% for ANN using simplex optimization for weight determination, 84% for CBR using GA for attribute weight selection) for building cost per square meter by using eight parameters available at the early design phase. Both models establish a methodology that can provide an economical and rapid means of cost prediction for the structural system of future building design processes.
3. Development of the models demonstrated the practicality of using spreadsheets in developing ANN and CBR for use in construction management. A spreadsheet simulation of an artificial neural network model developed by Hegazy and Ayed (1998) was the motivation of the investigation. The use of spreadsheets and development of ANN and CBR models in Excel have brought several benefits to the development process and presumably to the end user. This also indicates that the nature of the model development process in this study actually makes a unique difference in ML employment. It was possible to simulate the ANN and CBR processes in a transparent form, and further optimize them using spreadsheet availabilities.
4. The Excel simulations of ANN and CBR present these models as viable tools for use in construction by adjusting the developed templates to other applications. ANN and CBR Excel templates can be modified, populated with different sets of data and used in other areas of building construction such as quality, productivity, constructability, value engineering, scheduling, etc.
5. Spreadsheet programs have been among the most easy-to-use software programs that include powerful data management capabilities, since their introduction in early 1980s. Therefore, the use of spreadsheets in construction has been customary to many practitioners. Furthermore, users can also select among many add-in modules available on the market to extend spreadsheet capabilities. This study has used Solver and Evolver add-ins to Microsoft Excel to

improve weight generation abilities of ANN or CBR models. Additionally, the weights generated by other commercial softwares (Esteem and See5) have been easily plugged into these Excel models, which have facilitated integration and evaluation of different methods.

6. ANN learns from examples. The performance of an ANN model of cost prediction strongly depends on the quality and the quantity of examples. The more examples there are, the less the prediction error is. Thus, to study modeling and prediction methods in ANN, and construct an accurate prediction model of building costs, there is a need for reliable, highquality, full-scale cost data of buildings of various types and conditions. Though the data selected for this model were limited in scope, the results are encouraging for further research of expanded data sets.

7. CBR prediction model also depends on the examples in its casebase. CBR cost model performed well despite the fact that the number of cases in the casebase was small and the output attribute was not binary. Both the CBR prediction and the GA optimization of CBR-GA model suffered from the fact that not many of the 29 cases considered in the study had input attributes and outputs that were close to each other. The likelihood of seeing stronger similarities is much higher if the number of cases is substantially higher than 29.

8. ANN was used in this study to develop a prediction model where its connection weights were determined by three different approaches, namely simplex optimization, back propagation training and GA. Based on this experimentation; the simplex optimization produced the best ANN model. CBR was used in this study to develop a prediction model where attribute weights were generated by means of six different techniques, namely feature counting, gradient descent, GA and three methods of induction decision trees (ID3). The results indicated that GA-augmented CBR performed better than CBR used in association with the other techniques. Despite the limitation of data cited above, the study is of benefit to researchers as it illustrates the importance of weights as variables in the performance of both ANN and CBR prediction tools. It also indicates that it is worth

experimenting with different weight generation methods rather than being confined by the standard methodologies provided by ANN and CBR software.

9. The findings of ANN weight determination approaches show that GA optimization did not generate ANN predictions as strong as the simplex optimization method. The results of the ANN models using three different approaches for determining weights are presented in Table 4.2. The best overall model is the one produced by Excel Solver, providing superior performance on both the training and test cases. While back-propagation training produced a network with small errors on the training cases, it behaved relatively poor on the test cases. GA, on the other hand, did not produce good results probably because of its random selection of the generated population. Despite the consistent performance of the GA's model over the training and test cases, it exhibits a higher overall error. Therefore, it can be concluded that the networks of simplex optimization and back-propagation training are most suited to the present case study.

10. The findings of the CBR weight generation methods show that feature counting + CBR did not generate predictions that are as strong as the prediction generated by the GA + CBR because feature counting assigns equal weights to the attributes and therefore does not take into account the differences in importance of the attributes even though it is likely that such differences existed in the particular cases used in the study. But it was surprising to see that feature counting + CBR performed better than gradient descent + CBR. After all, gradient descent is a well established technique that is routinely used in CBR systems (e.g., Esteem 1996). While geometric descent was found to be more effective than arithmetic descent, the gradient descent experiments were conducted by using the default values of the parameters as recommended by Esteem (1996). Exploring the use of values other than the default values could possibly improve the performance of the gradient descent method, and in turn improve the predictions generated by gradient descent + CBR.

11. The findings of the CBR weight generation methods using ID3 methods show that info-top + CBR performed well considering the other decision tree methods, namely binary-dtree and info-dtree. All of the ID3 weight generation methods (binary-dtree, info-top, and info-dtree) and the CBR prediction suffered from the fact that not many of the 29 cases considered in the study had outputs that were close to each other. More consistent outputs could have resulted in splitting the cases into fewer classes in Table 4.4 and consequently producing smaller prediction errors in Table 4.3.
12. Also, while the CBR study concentrated on optimizing attribute weights, improving attribute selection can also be explored by using GA (Jarmulak and Craw 1999, Jarmulak et al. 2000).

Conclusions mainly cover methodological contributions that include the development of ANN and CBR Excel models and their testing results of cost data. ANN and CBR spreadsheet simulations integrated and enhanced by different methods, some not available in commercial softwares yet, can be extended beyond the specific cost problem addressed in this dissertation.

REFERENCES

Aamodt, A. and Plaza, E. 1994. "Case-based Reasoning: Foundational Issues, Methodological Variations, and System Approaches," *AICOM.* Vol.7, No.1, pp. 39-59.

Adeli, H. and Wu, M. 1998. "Regularization Neural Network for Construction Cost Estimation," *Journal of Construction Engineering and Management*. Vol.124, No.1, pp.18-24.

Adeli, H. 2001. "Neural Networks in Civil Engineering: 1989-2000," *Computer Aided Civil and Infrastructure Engineering*. Vol.16, pp.126-142.

Al-Tabtabai, H. and Alex, A.P. 2000. "Modeling the Cost of Political Risk in International Construction Projects," *Project Management Journal*. Vol.31, No.3, pp.4-13.

Al-Tabtabai, H., Alex, A.P., Tantash, M. 1999. "Preliminary Cost Estimation of Highway Construction Using Neural Networks," *Cost Engineering*. Vol.41, No.3, pp.19-24.

Arditi, D. and Tokdemir, O. B. 1999a. "Using Case-Based Reasoning to Predict the Outcome of Construction Litigation," Computer-*Aided Civil and Infrastructure Engineering*. Vol. 14. pp. 385-393.

Arditi, D. and Tokdemir, O. B. 1999b. "Comparison of Case-Based Reasoning and Artificial Neural Networks," *Journal of Computing in Civil Engineering*. Vol.13, No.3, pp. 162- 169.

Barrie, D.S. and Paulson, B.C. 1992. *Professional Construction Engineering and Management*, (McGraw-Hill, New York, NY).

Bode, J. 1998. "Neural Networks for Cost Estimation," *Cost Engineering*. Vol.40, No.1, pp.25-30.

Barrow, H. 1996. "Connectionism and Neural Networks," in *Artificial Intelligence: Handbook of Perception and Cognition*, edited by M.A. Boden (2nd edition, Academic Press, San Diego, CA), pp. 135-155.

Cardie, C. 1993. "Using Decision Trees to Improve Case-Based Learning," Proceedings of the Tenth International Conference on Machine Learning, ICML'93, University of Massachusetts, Amherst, MA, Morgan Kaufmann Publishers Inc., San Francisco, CA, pp. 25-32.

Carr, R.I. 1989. "Cost Estimating Principles," *Journal of Construction Engineering and Management.* Vol.115, No.4, pp.545-551.

CII. 1998. “Improving Early Estimates,” *Construction Industry Institute.* University of Texas at Austin, TX, USA.

Creese, R.C. and Li, L. 1995. “Cost Estimation of Timber Bridges Using Neural Networks,” *Cost Engineering*. Vol.37, No.5, pp.17-22.

Danyluk, A. 2004. “Learning Decision Trees.” and “Decision Trees on Real Problems.” Lecture Notes of CSCI 108 Artificial Intelligence: Image and Reality Course, Williams College, Department of Computer Science, Williamstown, MA.

Doğan, S.Z. and Günaydın, H.M. 2003. “Applications of Artificial Neural Networks and Their Potential Uses for Building Construction Industry: A Review,” Proceedings of the 9th EuropIA International Conference on E-Activities and Intelligent Support in Design and the Built Environment, İstanbul, edited by B. Tuncer, S. S. Ozsariyildiz and S. Sariyildiz, pp.79-89.

Doğan, S.Z., Arditi, D. and Günaydın, H.M. 2005a. “CBR Model for Early Cost Prediction,” *Journal of Construction Engineering and Management*. Under review.

Doğan, S.Z., Arditi, D. and Günaydın, H.M. 2005b. “Using Decision Trees for Determining Attribute Weights in a Case-Based Model of Early Cost Prediction,” *Journal of Construction Engineering and Management*. Under review.

Doğan, S.Z., Arditi, D. and Günaydın, H.M. 2005c. “GA Destekli VTG ile YSA’nın Elektronik Tablo Simülasyonlarının Karşılaştırılması,” Proceedings of Third National Construction Management Congress (3. Yapı İşletmesi Kongresi), İzmir, Turkey, (29 September – 30 September), pp.286-295.

Emsley, M.W., Lowe, D.J., Duff, A.R., Harding, A., Hickson, A. 2002. “Data Modeling and the Application of a Neural Network Approach to the Prediction of Total Construction Costs,” *Construction Management Economics*. Vol.20, pp.465-472.

Esteem 1.4. 1996. *Case based reasoning development tool*. Esteem Software, San Mateo, California.

Evolver 4.0. 1998. *Excel Reference Manual*. Palisade Corp., Newfield, NY.

Excel. 2003. *Microsoft Excel Documentation.* Microsoft Corporation.

Fausett, L. 1994. *Fundamentals of Neural Networks*, (Prentice Hall, Englewood Cliffs, NJ).

Feery, D. and Brandon, P.S. 1984. *Cost Planning of Buildings,* (Collins, London, UK).

Francone, F. D. 1999. “AIM learning, Adaptive, Real Time, Control Technologies,” unpublished manuscript available at the web site (27/08/2005) http://www.aimlearning.com/Process%20Control%20White%20Paper.pdf

Friedman, J. H. 2003. “Recent Advances in Predictive (Machine) Learning.” Proceedings of Statistical Problems in Particle Physics, Astrophysics and

Cosmology, SLAC Stanford, California, (November 2003), manuscript available at Friedman's web site (27/08/2005) http://wwwstat.stanford.edu/~jhf/ftp/machine.pdf.

Gupta, U.G. 1994. "How Case-Based Reasoning Solves New Problems," *Interfaces*. Vol.24, No.6, pp.110-119.

Günaydın, H.M. and Doğan, S.Z. 2004. "A Neural Network Approach for Early Cost Estimation of Structural Systems of Buildings," *International Journal of Project Management.* Vol.22, No.7, pp. 595-602.

Harding, A., Lowe, D., Hickson, A., Emsley, M. and Duff, R. 2000. Implementation of a neural network model for the comparison of the cost of different procurement approaches. Paper presented to CIB W92 Procurement System Symposium. Santiago, (April), Chile, pp.24–27.

Hastie, T. 2004. University of Stanford web site, 11/11/2004. *http://www.stanford.edu /class/stats315a/.* ("Statistics 315B: Modern Applied Statistics: Elements of Statistical Learning," introductory course notes).

Hastie, T., Tibshirani, R. and Friedman, J.H. 2001. *The Elements of Statistical Learning*: *Data Mining, Inference, and Prediction,* (Springer-Verlag, New York, NY).

Haykin, S., 1994. *Neural Networks: A Comprehensive Foundation*, (Macmillan, NewYork).

Hegazy, T. and Ayed, A. 1998. "Neural Network Model for Parametric Cost Estimation of Highway Projects," *Journal of Construction Engineering and Management*. Vol.124, No.3, pp.210-225.

Hegazy, T., Moselhi, O., and Fazio, P. 1994. "Developing Practical Neural Network Applications Using Back-Propagation," *Microcomputers in Civil Engineering*. Vol.9, No.2, pp.145-159.

Heery, G.T., 1975. *Time, Cost, and Architecture,* (McGraw-Hill, New York.)

Hunt Jr., W.D. 1967. *Creative Control of Building Costs,* (McGraw-Hill, New York.)

Induce-It. 2000. *User Manual.* Inductive Solutions, Inc., New York, NY.

Jarmulak, J. and Craw, S. 1999. "Genetic Algorithms for Feature Selection and Weighting," in *IJCAI-99 Workshop on Automating the Construction of Case-Based Reasoners ,* Stockholm, Sweden, (2 August 1999) edited by S. S. Anand, A. Aamodt, and D. W. Aha, pp. 28-33.

Jarmulak, J., Craw, S. and Rowe, R. 2000. "Genetic Algorithms to Optimize CBR Retrieval," in EWCBR 2000, LNAI 1898, edited by E. Blanzeri and L. Portinale (Springer-Verlag, Berlin Heidelberg), pp. 136-147.

Kalogirou, S.A. 1999. “Applications of Artificial Neural Networks in Energy Systems: A Review,” *Energy Conversion and Management*. Vol.40, No.3, pp.1073-1087.

Kalogirou, S.A. 2001. “Artificial Neural Networks in Renewable Energy Systems Applications: A Review,” *Renewable and Sustainable Energ Reviews*. Vol.5, No.4, pp.373-401.

Karshenas, S. 1984. “Predesign Cost Estimating Method for Multistory Buildings,” *Journal of Construction Engineering and Management.* Vol.111, No.1, pp.79-99.

Kasravi, K. 1994. ‘‘Understanding Knowledge-Based CAD/CAM.’’ *Journal of Computer Aided Engineering.* Vol.13, No.10, 72–78.

Kibler D. and Aha D.W. 1987. “Learning Representative Exemplars of Concepts: An Initial Case Study,” Proceedings of the Fourth International Workshop on Machine Learning, Irvine, CA, (June 1987), edited by P. Langley, Morgan Kaufmann, CA, pp. 24-30.

Kolodner, J.L. 1991. “Improving Human Decision Making Through Case-Based Decision Aiding,” *AI Magazine*. Vol.12, No.2, pp.52-68.

Kolodner, J.L. 1993. *Case based reasoning,* (Morgan Kaufmann Publishers, Inc., San Mateo, CA).

Ling, C. X., Parry J. J. and Wang, H. 1997. “Setting Attribute Weights for Nearest Neighbor Learning Algorithms Using C4.5,” *International Journal of Pattern Recognition and Artificial Intelligence*. Vol.11, No.3, pp. 405 - 415.

Melin, J.B. 1994. “Parametric Estimation,” *Cost Engineering*. Vol.36, No.1, pp.19-24.

Mukherjee, A. and Deshpande, J.M. 1995. “Modeling Initial Design Process Using Artificial Neural Networks,” *Journal of Computing in Civil Engineering*. Vol.9, No.3. pp.194-200.

NeuroShell Trader Professional. 2002. *Tutorial,* WardSytems Group, Inc., Frederick, MD.

NeuroSolutions. 2002. NeuroSolutions Tool for Excel, NeuroDimensions, Inc. Gainesville, FL.

Orhon, İ., Sey, Y., Aral, N., Giritli, H., Sözen, Z. 1986-1987 fall semester lecture notes. Istanbul Technical University.

Paek, J.H. 1994. “Contractor Risks in Conceptual Estimating,” *Cost Engineering*. Vol.36, No.12, pp.19-22

Quinlan, J.R. 1996. “Bagging, Boosting, and C4.5,” Proceedings, Fourteenth National Conference on Artificial Intelligence, manuscript available at web address http://www.cse.unsw.edu.au/~quinlan/q.aaai96.ps.

Rafiq, M.Y., Bugmann, G., Easterbrook, D.J. 1998. "Artificial Neural Networks for Modeling Some of the Activities of the Conceptual Stage of the Design Process," *Proceedings of International Computing Congress on Computing in Civil Engineering*, edited by K.C.P Wang (Boston, Massachusetts), pp.631-643.

Rafiq, M.Y, Bugmann, G. and Easterbrook, D.J. 2001. "Neural Network Design for Engineering Applications," *Computers and Structures*. Vol. 79, pp. 1541-1552.

ReCall. 1993. *A Case Based Reasoning Shell.* Isoft, France.

Reich, Y. 1997. ''Machine Learning Techniques for Civil Engineering Problems.'' *Microcomputers in Civil Engineering.* Vol. 12, No. 4, pp. 295-310.

Riesbeck, C.K. and Schank, R.C. 1989. *Inside case-based reasoning,* (Lawrence Erlbaum Associates, Hillsdale, NJ).

Rumelhart, D.E., Hinton, G.E., Williams, J.R. 1986. "Learning Representations by Backpropagation Errors," *Nature*. Vol. 323, pp. 533-536.

Saner, C. 1993. A Proposal for Cost-Estimation for Structural Systems of 4–8 Storey Residential Buildings. MSc. Thesis, Istanbul Technical University.

Schank, R.D. 1982. *Dynamic Memory; a Theory of Reminding and Learning in Computers and People*, (Cambridge University Press, New York, NY).

See5/C5.0. 1997. *Data Mining Tools Manual.* Rulequest, Australia.

Setyawati, B.R, Sahirman, S., Creese, R.C. 2002. "Neural Networks for Cost Estimation," *AACE International Transactions*, ABI/INFORM Global: EST.13.1–EST.13.9.

Seyyar, B. 2000. Computer Aided Cost Estimation Systems During Building Design Process. MSc. Thesis, Istanbul Technical University.

Shi, J. J. 2000. "Reducing Prediction Error by Transforming Input Data for Neural Networks," *Journal of Computing in Civil Engineering*. Vol. 14 No. 2, pp. 109-116.

Shin, K. and Han, I. 1999. "Case Based Reasoning Supported by Genetic Algorithms for Corporate Bond Rating," *Expert Systems with Applications*. Vol. 16, No. 2, pp.85-95.

Shtub, A., Versano, R. 1999. "Estimating the Cost of Steel Pipe Bending, a Comparison Between Neural Networks and Regression Analysis," *International Journal of Production Economics.* Vol.62, No.3, pp.201-207.

Smith, A.E., Mason, A.K. 1997. "Cost Estimation Predictive Modeling: Regression Versus Neural Network," *The Engineering Economist*. Vol.42, No.2, pp.137-161.

Squeira, I. 1999a. Neural Network-Based Cost Estimating, Master's Thesis. Department of Building, Civil and Environmental Engineering, Concordia University, Montreal, Quebec, Canada.

Siqueira, I. 1999b. "Automated Cost Estimating Systems Using Neural Networks," *Project Management Journal.* Vol.30, No.1, pp.11-18.

State Institute of Statistics, Construction Permits for the Year 2003. Available from: www.die.gov.tr/english/SONIST/INSAAT/050903g.htm; last accessed October 2003.

Stottler, R.H. 1994. "CBR for Cost and Sales Prediction." *AI Expert*. Vol. August, pp. 25-33.

U.S. Department of Defense. 1995. "Parametric Cost Estimating Handbook," *Department of Defense, United States of America.* Arlington, VA, USA.

Watson, I. and Marir, F. (1994a). "Case Based Reasoning: A Review," *The Knowledge Engineering Review*. Vol.9, No.4, pp.327-354.

Watson, I. and Marir, F. (1994b). "Case Based Reasoning: A Categorized Bibliography," *The Knowledge Engineering Review*. Vol.9, No.4, *manuscript available at the web address http://www.salford.ac.uk/survey/staff/IWatson/ cbrefs.htm*.

Yau, N.J. and Yang, J.B. 1998. "Case based Reasoning in Construction Management." *Computer-Aided Civil and Infrastructure Engineering.* Vol.13, pp.143-150.

Yaylagül, N. 1994. Bina Yapımında Simülasyon Yaklaşımıyla Maliyet Tahmini (Cost Estimation via Simulation Approach in Building Construction). MSc. Thesis, Istanbul Technical University.

Zhang, Y.F. and Fuh, J.Y.H. 1998 ."A Neural Network Approach for Early Cost Estimation of Packaging Products," *Computers and Industrial Engineering*. Vol.34, No.2, pp.433-450.

APPENDIX A

TOOLS FOR CASE-BASED REASONING

Vendors and service providers are (name of company followed by name of tool and URL where available):

- Atlantis Aerospace Corporation and later Case Bank Support Systems Inc, **Spotlight**, http://www.casebank.com/products/spotlight.asp

Phil D'Eon, Chairman and Chief Executive Officer**,** is a founder of CaseBank and originator of the SpotLight concept. In 1978, he co-founded Atlantis Aerospace Corporation, and for over 20 years developed a successful international business in maintenance and flight simulators for the aerospace industry. It was there that he originated the SpotLight concept in 1995. In 1998, he purchased the SpotLight technology from Atlantis and founded CaseBank. He continues today to guide the innovation of CaseBank's case-based reasoning technology and pioneering new applications.

- The University of Wales, Aberystwyth, **Caspian**,

http://www.aber.ac.uk/compsci/Research/mbsg/cbrprojects/getting_caspian.shtml
This is a publicly available CBR shell built at Aberystwyth.

- Cognitive Systems, Inc., **ReMind**: Case-based Reasoning Development Shell,

Cognitive Systems Inc. ceased trading in 1996. ReMind may still be available from other suppliers.

- Esteem Software, Inc. and later SHAI: Stottler-Henke Associates, Inc
  ESTEEM: Case-based Reasoning Shell,
  http://www.stottlerhenke.com/solutions/decision_support/esteem.htm

ESTEEM enables people to develop case-based reasoning applications without programming. ESTEEM was marketed by Esteem Software Incorporated from 1991 to 2001. ESTEEM is described as being "a good tool for people interested in exploring the potential of CBR within their organizations." Research and academic institutions can request a free copy of this unsupported software by sending email to info@stottlerhenke.com.

- Inductive Solutions, Inc., **Induce-It**, http://www.inductive.com/softcase.htm

Induce-It is an Excel-based case-based reasoning system. It creates case-based reasoning systems from Microsoft Excel spreadsheet databases. Induce-It searches a case database based on similarity metrics. Case-based are adapted from the closest matching cases, ranked by case score, and displayed to users in a sorted list. f you know how to use a spreadsheet, then you know how to use Induce-It. Student version is $85
for 60 day license and you can download it from
http://www.inductive.com/download.htm

- ALICE d'ISoft**,** ReCall, http://www.alice-soft.com/html/prod_recall.htm

ReCall is a CBR toolkit, which helps you re-use your corporate knowledge. ReCall is also available as a set of libraries for developers. You can run an on-line demo of ALICE d'ISoft. You can download the ALICE d'ISoft demonstration. You can fill in a form to receive an evaluation version of the product.

- Simon Fraser University, **Case Advisor 2.1**,
  http://www.cs.sfu.ca/research/groups/CBR/

Case Advisor is a PC-based problem diagnosis and resolution system which allows an organization to retrieve solutions from a "knowledge database" to solve customer problems. CaseAdvisor 4 PC Version is given out free for non-commercial purposes.

You can download the installation program at, Download Case Advisor 4.12: http://www.cs.sfu.ca/~isa/caseadvisor/download/, Case Advisor Screen Demos:

http://www.cs.sfu.ca/~isa/caseadvisor/screendemo/index.html. After viewing the demo, you may receive a 30-day evaluation copy of CaseAdvisor Software. If interested in
receiving the evaluation copy, contact Peter Leung at SoundLogic peterl@soundlogic.net: 604-291-9989 x3022 or Dr. Qiang Yang: 604-291-5415.

- TreeTools**,** HELPDESK-3, http://www.treetools.com.br/ (unfortunately this site is in Portuguese only)

HELPDESK-3 from the Brazilian company TreeTools is a CBR tool designed to automated help desk. It uses heuristic search to retrieve cases and can handle natural language problem description.

- University of Auckland, Department of Computer Science, CS760 Datamining & Machine Learning, AIAI CBR-Tool, http://www.cs.auckland.ac.nz/~ian/760/

This tool lets you explore various features of a CBR tool including adaptation. It can be downloaded. The software is free for academic use.

Names of other vendors:

- Inference Corporation - k-commerce (formerly called CBR3 or CBR Express)
- IET-Intelligent Electronics - TechMate
- Intellix - KnowMan
- Sententia Software Inc. - CASE Advisor & Case Advisor Webserver
- ServiceSoft - Knowledge Builder & Web Adviser
- tecInno GmbH - CBR-Works and Inference's k-commerce
- Web*presence* Technology - The RapidReasoner
- Astea International - Case-1

APPENDIX B

BOOSTED DECISION TREES

Introduction

In many problem domains, combining the predictions of several models often results in a model with improved predictive performance. The trend toward model mixing had a resurgence in economics (Bates and Granger 1969), has increased in the machine learning community. Boosting is one of such method that is an addition to the class of model mixing procedures. This study was conducted to see if boosting concept combined with decision trees could be used for better cost estimation results than the ones obtained by ANN and CBR models. Thus, this report provides an introduction to boosting algorithm and decision trees, presents an application of boosted decision trees (BDT) to cost estimation, and discusses the prediction results provided by BDT model.

Boosting Algorithm

Given a training set of data, a learning algorithm will generate a rule that classifies the data. This rule may or may not be accurate, depending on the quality of the learning algorithm and the inherent difficulty of the particular classification task. If the rule is even slightly better than random guessing, then the learning algorithm has found some structure in the data to achieve this advantage. Boosting is a method that boosts the accuracy of the learning algorithm by making the most of this advantage. Boosting uses the learning algorithm routinely in order to produce a prediction rule that is guaranteed to be highly accurate on the training set. Boosting works by running the learning algorithm on the training set multiple times, each time focusing on different training examples. After the boosting process is finished, the rules that were output by the learner are combined into a single prediction rule which is provably accurate on the training set. This combined rule is then verified for its accuracy on the test set.

Boosting has its roots in a theoretical framework for studying machine learning called the Probably Approximately Correct (PAC) learning model, due to Valiant (Quinlan, 1996); see Kearns and Vazirani (1994) for a good introduction to this model. Kearns and Valiant (1988, 1994) were the first to pose the question of whether a "weak" learning algorithm which performs just slightly better than random guessing in the PAC

model can be "boosted" into an arbitrarily accurate "strong" learning algorithm. Schapire (1990) came up with the first provable polynomial-time boosting algorithm in 1989. Later, Freund (1995) developed a much more efficient boosting algorithm which, although optimal in a certain sense, nevertheless suffered from certain practical drawbacks. The first experiments with these early boosting algorithms were carried out by Drucker et al. (1993) on an Optical Character Recognition (OCR) task. The AdaBoost algorithm, introduced in 1995 by Freund and Schapire (1999), solved many of the practical difficulties of the earlier boosting algorithms, and is the one used in this study.

The AdaBoost algorithm was a breakthrough. When the first boosting algorithms were invented they received a small amount of attention from the experimental machine learning community (Drucker et al. 1993). Then the AdaBoost algorithm arrived with its many desirable properties: a theoretical derivation and analysis, fast running time, and simple implementation. These properties attracted machine learning researchers who began experimenting with the algorithm. All of the experimental studies showed that AdaBoost almost always improves the performance of various base learning algorithms, often by a dramatic amount (Drucker et al. 1993).

Decision Trees

In this section, the application of boosting to one kind of base learning algorithm that outputs decision tree classifiers is discussed. Experiments with the AdaBoost algorithm usually apply it to classification problems. Recall that a classification problem is specified by a space X of instances and a space Y of labels, where each instance x is assigned a label y according to an unknown labeling function c: $X \rightarrow Y$. We assume that the label space Y is finite. The input to a base learning algorithm is a set of training examples "$(x_1;y_1),\ldots,(x_m;y_m)$", where it is assumed that y_i is the correct label of instance x_i (i.e., $y_i = c(x_i)$). The goal of the algorithm is to output a classifier $h:X \rightarrow Y$ that closely approximates the unknown function c.

The first experiments with AdaBoost (Drucker and Cortes, 1996; Freund and Schapire, 1996; Quinlan, 1996) used it to improve the performance of algorithms that generate decision trees, which are defined as follows. Suppose each instance $x \in X$ is represented as a vector of n attributes "$a_i,\ldots,a_n$" that take on either discrete or continuous values. For example, an attribute vector that represents human physical characteristics is "height, weight, hair color, eye color, skin color". The values of these

attributes for a particular person might be “1.85 m, 70.5 kg, black, dark brown, tan.” A decision tree is a hierarchical classifier that classifies instances according to the values of their attributes. Each non-leaf node of the decision tree has an associated attribute a (one of the a_i's) and a value v (one of the possible values of a). Each non-leaf node has three children designated as “yes”, “no”, and “missing.” Each leaf node u has an associated label $y \in Y$.

A one node decision tree, called a stump, consists of one internal node and three leaves. Consider a stump T_1 whose internal node compares the value of attribute a to value v. T_1 classifies instance x as follows. Let x.a be the value of attribute a of x. If a is a discrete-valued attribute then

- if x.a = v then T_1 assigns x the label associated with the “yes” leaf.
- if x.a $\neq$ v then T_1 assigns x the label associated with the “no” leaf.
- if x.a is undefined, meaning x is missing a value for attribute a, then T_1 assigns x the label associated with the “missing" leaf.

If instead a is a continuous-valued attribute, T_1 applies a threshold test (x.a > v) instead of an equality test.

A general decision tree T has many internal nodes with associated attributes. In order to classify instance x, T traces x along the path from the root to a leaf u according to the outcomes at every decision node; T assigns x the label associated with leaf u. A decision tree can be thought of as a partition of the instance space X into pair wise disjoint sets X_u whose union is X, where each X_u has an associated logic expression that expresses the attribute values of instances that fall in that set
(for example “eye color = blue and height < 1.25 m”).

The goal of a decision tree learning algorithm is to find a partition of X and an assignment of labels to each set of the partition that minimizes the number of mislabeled instances.

About See 5 (Boosted Decision Trees)

Freund and Schapire (1996) and Quinlan (1996) investigated the abilities of boosting to improve C4.5, a decision tree learning algorithm. When using C4.5 as the Base learner, Freund and Schapire's (1996) experiments revealed that on average, boosting improved the error rate of C4.5 by 24.8%. Quinlan (1996) found that boosting reduced C4.5's classification error by 15%. Drucker and Cortes (1996) also found that

AdaBoost was able to improve the performance of C4.5. They used AdaBoost to build ensembles of decision trees for optical character recognition (OCR) tasks. In each of their experiments, the boosted decision trees performed better than a single tree, sometimes reducing the error by a factor of four.

Experiments with the AdaBoost algorithm revealed that it is able to use a base-learning algorithm to produce a highly accurate prediction rule. AdaBoost usually improves the base learner quite dramatically, with minimal extra computation costs (Valiant, 1997). Valiant (1997) praised AdaBoost for being an extremely simple algorithm that can get practitioners use in minutes.

See 5 is a successor of C4.5, which combines decision trees with AdaBoost algorithm. It was written by Quinlan in 1996. Because of their simplicity compared with other artificial intelligence systems, boosting was first experimented with decision trees to test if performance is enhanced by this plug-in algorithm. See 5 and TreeBoost (DTreg) are the available software tools that integrate boosting with decision trees. The idea is to generate several classifiers of decision trees rather than just one. When a new case is to be classified, each classifier votes for its predicted class and the votes are counted to determine the final class.

The first step in generating several classifiers from a single dataset involves constructing a single decision tree using the training data. Once the results of this classifier are obtained, the data on which it has made mistakes are determined. Then, the second classifier is constructed by paying more attention to the wrong predicted data in an attempt to get them right. Consequently, the second classifier will generally be different from the first. It also will make errors on some data, and these will become the focus of attention during the construction of the third classifier. This process continues for a pre-determined number of iterations (Arditi and Pulket 2004, Pulket 2001).

The Boost option with *x* trials instructs See5 to construct up to *x* classifiers in this manner. Although constructing multiple classifiers requires more computation than building a single classifier, the prediction results are generally much better.

Procedure of Preparing Data for See 5

This section shows how to prepare data files for See 5 and the procedure of running the system. The data involved for this application belongs to residential buildings in Turkey. The objective was to construct boosted decision trees to predict the

unit structural cost of these buildings. Below are the attributes and information related to some cases:

Attribute	**Case 1**	**Case 2**	**Case 3**
The total area of the building (m^2)	675	1425	330
The ratio of the typical floor area to the total area of the building	0.2	0.2	0.2
The ratio of the ground floor area to the total area of the building	0.182	0.2	0.2
The number of floors	6	6	6
The console direction of the building	oneway	nocons	nocons
The location of the core of the building	sides	sides	sides
The floor type of the building	precast	reinforced	reinforced
The foundation system of the building	wall	wall	wall
The cost of the structural system per m2	2	3	1

Abbreviation:

oneway = one-way console

nocons = no consoles

sides = at the sides

middle = in the middle

precast = precast concrete structural units

reinforced = reinforced concrete floor systems

The unit structural costs of cases are classified into 13 classes. Class 1 represents the unit cost of the structural system falling between $30 and $40. Class 2 represents the unit cost range between $40 and $50. The classification goes on like that up to the last class 13 which represents the unit cost range between $150 and $160. Each case belongs to one of a small number of these mutually exclusive classes. Properties of each case that may be relevant to its outcome are provided. There are 8 attributes in the problem and the system investigates how to predict the unit structural cost of the

building from the values of these attributes. See 5 does this by constructing a classifier that makes this prediction.

Two files are essential for all See 5 applications and there are three further optional files (See 5 Tutorial). The first essential file is the "names" file that describes the attributes and classes. The names file for this data set is as follows:

Names File

```
the cost of the structural system per m2 :   | the target attribute

the total area of the building (m2) :        continuous.

the ratio of the typical floor area
to the total area of the building :          continuous.

the ratio of the ground floor area
to the total area of the building :          continuous.

the number of floors :                       4,5,6,7,8.

the console direction of the building :      nocons, oneway.

the location of the core of the building :   sides, middle.

the floor type of the building :             precast, reinforced.

the foundation system of the building :      pier, wall, slab.

the cost of the structural system per m2:    1,2,3,4,5,6,7,8,9,
                                             10,11,12,13.
```

The second file is the application's data file which provides information on the cases for See 5 in order to extract patterns. The entry for each case gives the values of all attributes available for that case. Commas separate values. The data file for this study is as follows: (The data for testing is separated by a horizontal line)

Data File

```
675,0.2,0.182,6,oneway,sides,precast,wall,2
1425,0.2,0.2,6,nocons,sides,reinforced,wall,3
330,0.2,0.2,6,nocons,sides,reinforced,wall,1
2025,0.14,0.13,5,nocons,middle,precast,wall,4
1670,0.16,0.16,4,oneway,sides,precast,pier,5
2082,0.16,0.3,6,oneway,sides,precast,wall,3
3484,0.07,0.07,6,nocons,middle,reinforced,wall,4
1364,0.25,0.23,6,nocons,sides,reinforced,pier,5
1568,0.26,0.25,6,nocons,middle,reinforced,slab,6
⋮
569,0.16,0.14,6,oneway,sides,reinforced,wall,2
1156,0.13,0.095,6,oneway,middle,reinforced,wall,2
1146,0.202,0.19,5,nocons,sides,reinforced,slab,9
⋮
2528,0.13,0.096,8,oneway,middle,reinforced,wall,2
```

Once the names, data and other optional files have been set up, everything is ready for See 5 to construct classifiers. Several options affect the type classifier that See 5 produces and the way that it is constructed. The "Construct Classifier" button on the toolbar displays a dialog box that sets out these classifier construction options.

Discussion and Results of See 5 Application for the Prediction of Cost Data

See 5 constructed the classifiers for the same data set that was also used in the ANN and CBR models. The Boosting option was set to 10 trials. However, the boosting was reduced to three trials since the last classifier constructed by See 5 was very inaccurate. Indeed, the system abandoned boosting since there were too few classifiers. The first experiments showed that the decision tree model of See 5 couldn't be boosted. This resulted in very poor prediction rates of 47.6%, 52.4% and 61.9% on the training set. Since the training of the data was not successful, the testing evaluations were poorer. Substantial manipulation of the data is required for BDT application in order to enable BDT to make accurate predictions. For increasing the prediction accuracy of the system, altering the size of the classes and dividing the target values into less number of classes are required. This modification does not serve to the practical goal of this dissertation since exactness in cost values will most probably be affected negatively by this way. However, this will give us some idea about the application's capabilities. The details of these modifications are reported below. Although BDT model results are tried to be improved by carefully manipulating the data, no valid and consistent improvement in prediction accuracy could be achieved.

Experiments on the Dataset

- In the first part of the study, 29 cases used in the ANN and CBR models were entered. However, the original data were in a format that created compatibility problems with See5. Therefore, the original data needed adjustment to conform the requirements of See5 and target attribute values which were the unit structural cost values were classified into 13 classes:
- See5 using the decision tree it constructed for the cost data with no boosting produced a prediction rate of 62.5% on the training set.
- When the boosting option was set to 10 times, the system couldn't be boosted. Boosting was reduced to 3 trials since the last classifier was reported to be very

Table 1. Identified classes for the target attribute

Class No	*Cost per meter*
1	$30 – $40
2	$40 – $50
3	$50 – $60
4	$60 – $70
5	$70 – $80
6	$80 – $90
7	$90 – $100
8	$100 – $110
9	$110 – $120
10	$120 – $130
11	$130 – $140
12	$140 – $150
13	$150 – $160

inaccurate and boosting was abandoned because of too few classifiers.

- Then the attributes were reduced from 8 to 2. These 2 attributes were selected to be the ones found to have the greatest impact on the ANN model by the sensitivity analysis (Günaydın and Doğan 2004). In the ANN model, the ratio of typical floor area to the total area of the building and the ratio of the footprint area to the total area of the building were found to be the most effective design parameters. However, only using these two design parameters reduced the average prediction accuracy of the ANN model from 93% to 90%. This finding might suggest for the ANN model that even the small clues (i.e. attributes) could enhance the model's prediction capability. For the BDT model (with no boosting), the prediction rate with these 2 attributes were 33.5% on the training set and 50% on the testing set.
- When the first decision tree model with 8 attributes was considered, it was seen that the attribute "foundation system" is the main classifier for the tree constructed. Therefore, this attribute was added and the

experiment was repeated with 3 attributes this time. The results (with no boosting) were 47.6% on the training set and 75% on the testing set.

- Finally, the decision tree model was run with 1 attribute of "foundation system." The prediction accuracy was 47.6% on the training and 25% on the testing set.
- Target attribute values which are the unit structural cost values were classified into 13 classes in the experiments. Using classes for cost prediction in this case means that even the prediction accuracy on the testing set is 100%, the predictor will still have an error rate of 33.3% since a predicted class will still have a prediction range (i.e. predicted class 1 would mean a cost value between $30 and $40). Thus, the aim already became 100% accuracy on the testing test in order for the model to be worthy. This led us to try the reduction of the classes. Since only 1 case belongs to 12^{th} class and again one case belongs to 10^{th} class and relatively less cases belong to 9^{th}, 10^{th} and 11^{th} classes considering the first 8; the classes after 8^{th} has been eliminated. Then the experiments were carried out with 8 classes. The results were still not promising. The system put out a prediction rate of 55% on training set and 0% on the testing set. Therefore, the study was extended into the next phase with the objective to improve the prediction rate.
- During experimentation, it was observed that some cases cause the accuracy of the training and testing to decline. Therefore, these cases were eliminated and the number of training cases was cut down to 19 and the number of testing cases was cut down to 2 from the original whole dataset of 29 cases.
- Using the new training set of 19 cases, 94.7% prediction accuracy on the training set and 100% prediction accuracy on the testing set were achieved by 10 boosting trials. This means the system was able to find the correct class although each class suggests a cost range. However, testing with 2 cases and using 19 training cases was already a failure for a prediction problem.

References

Arditi, D. and Pulket, T. (2004). "Predicting the outcome of construction litigation using boosted decision trees." Unpublished manuscript.

Bates, J. M. and Granger, C. W. J. (1969). "The combination of forecasts." *Operations Research Quarterly,* 20, 451–468.

Drucker, H., Schapire, R. E., and Simard, P. (1993). "Boosting performance in neural networks." *International Journal of Pattern Recognition and Artificial Intelligence,* 705-719.

Drucker, H. and Cortes, C. (1996). "Boosting decision trees." *Advances in Neural Information Processing Systems,* 8, 479-485.

DTREG. (2004). Download and technical information available at the web address: http://www.dtreg.com

Freund, Y. (1995). "Boosting a weak learning algorithm by majority." *Information and Computation,* 256-285.

Freund, Y. and Schapire, R. (1996). "Experiments with a new boosting algorithm." *Machine Learning: Proceedings of Thirteenth International Conference*, 148-156.

Freund, Y. and Schapire, R. E. (1999). "A short introduction to boosting." *Journal of Japanese Society of Artificial Intelligence,* 14(5), 771-780.

Kearns, M. J. and Valiant, L. G. (1988). "Learning Boolean formulae or finite automata is as hard as factoring." Technical Report TR -14-88, Harvard University Aiken Computation Laboratory.

Kearns, M. and Leslie, G. V. (1994). "Cryptographic limitations on learning boolean formulae and finite automata." *Journal of the Association for Computing Machinery,* 41(1), 67-95.

Kearns, M. J. and Vazirani, U. V. (1994). *An Introduction to Computational Learning Theory.* MIT Press, Cambridge, MA.

Pulket, T. (2001). "Predicting the outcome of construction claims using boosted decision trees." Thesis submitted for the degree of Master of Science in Civil Engineering in the Graduate College of the Illinois Institute of Technology.

Quinlan, J. R. (1996). "Bagging, boosting and C 4.5." *Proceedings of Fourteenth National Conference on Artificial Intelligence*, 725-730.

Rule Quest Research Pty Ltd. (1999). "See5: Informal Tutorial." *Unpublished Manuscript, available from the company homepage.* (http://www.rulequest.com)

Schapire, R. E. (1990). "The strength of weak learnability." *Machine Learning*, 197-227.

Valiant, L. G. (1997). "Learning is computational." Knuth Prize Lecture, In 38th Annual Symposium on Foundations of Compute

VITA

Sevgi Zeynep Doğan, daughter of Firuz and Kiper, was born on 14th November 1977 in İzmir. She attended Bornova Anatolian High School in İzmir, and was graduated in 1995 with the 3rd rank. She was admitted to Middle East Technical University in September 1995 and received her bachelor's degree in Architecture in June 1999 being an honour student. She was accepted to the Master of Science Program in Architecture in September 1999 and received her M.Sc. Degree in December 2000.

She is currently holding a research/teaching assistant position at İzmir Institute of Technology (Iztech) since December 2001. She has assisted building technology and science courses, basic design and architectural design studios at Iztech. From January 2004 to January 2005 she worked as a visiting research scholar in the Department of Civil and Architectural Engineering, Illinois Institute of Technology, Chicago, USA.

Her research interests are construction management technologies particularly machine learning / artificial intelligence implementations in construction problems. She has research papers in International Journal of Project Management and proceedings of EuropIA and national conference on Construction Management.

Her current e-mail address is sevgidogan@iyte.edu.tr or dogan@iit.edu and her web address is http://www.iyte.edu.tr/~sevgidogan/