IMPROVEMENT AND IMPLEMENTATION OF ANALOGY BASED METHOD FOR SOFTWARE PROJECT COST ESTIMATION

LI YAN-FU

(B. Eng), WUHAN UNIVERSITY

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF INDUSTRIAL AND SYSTEMS ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2009

Acknowledgements

First and foremost, I would like to record the deepest gratitude to my advisors, Prof. Xie Min and Prof. Goh Thong Ngee, whose patience, motivation, guidance and supports from the very beginning to the final stage of my PhD life enabled me to complete the research works and this thesis.

Besides my advisors, I would like to thank the professors who taught me lectures and gave me wise advices, the student colleagues who provided me a stimulating and fun environment, the laboratory technicians and secretaries who offered me great assistants in many different ways.

I wish to thank my wife and my best friends in NUS for helping me get through the difficult times, and for all the emotional support, entertainment, and caring they provided.

Last but not the least, I should present my full regards to my parents who bore me, raised me, and loved me.

To them I dedicate this thesis.

Yanfu Li

Table of Contents

Summary

Cost estimation is an important issue in project management. The effective application of project management methodologies often relies on accurate estimates of project cost. Cost estimation for software project is of particular importance as a large amount of the software projects suffer from serious budget overruns. Aiming at accurate cost estimation, several techniques have been proposed in the past decades. Analogy based estimation, which mimics the process of project managers making decisions and inherits the formal expressions of case based reasoning, is one of the most frequently studied methods.

However, analogy based estimation is often criticized for its relatively poor predictive accuracy, large computational expense, and intolerance to uncertain inputs. To alleviate these drawbacks, this thesis is devoted to improve the analogy based method from three aspects: accuracy, efficiency, and robustness.

A number of journal/conference papers have been published under this objective. The research works that have been done are grouped into four chapters (each chapter is focused on one component of analogy based estimation): chapter 3 summarizes the work on mutual information based feature selection technique for similarity function; chapter 4 presents the research on genetic algorithm based project selection method for historical database; chapter 5 presents the work on non-linear adjustment to solution function; chapter 6 presents the probabilistic model of analogy based estimation with focus on the number of nearest neighbors. The remaining chapters in this thesis, namely chapters 2 and 7, are the literature review and the conclusions and future works.

Research in chapters 3 to 5 aims to enhance analogy based estimation's accuracy. For instance, in chapter 5 the adjustment mechanism has been largely improved for a more accurate analogy based method. Efficiency is another important aspect of estimation performance. In chapter 3, our study on refining the historical dataset has achieved a significant reduction of unnecessary projects and therefore improved the efficiency of analogy based method. Moreover, in chapter 6 the study on probabilistic model lead to a more robust and reliable analogy based method tolerable to uncertain inputs.

The promising results show that this thesis makes significant contributions to the knowledge of analogy based software cost estimation in both the fields of software engineering and project management.

List of Tables

List of Figures

List of Abbreviations

ABE:	Analogy based estimation
ANN:	Artificial neural network
BABE:	Bootstrapped analogy based estimation
CART:	Classification and regression trees
CASE:	Computer-aided software engineering
FWABE:	Feature weighting for analogy based estimation
FWPSABE:	Simultaneous feature weighting and project selection for analogy based estimation
GABE:	Genetic algorithm optimized linear function adjusted analogy based estimation
KNNR:	K-nearest neighbor regression
LABE:	Linear function adjusted analogy based estimation
MdMRE:	Median Magnitude of Relative Error
MIABE:	Mutual information based features selection for analogy based estimation
MMRE:	Mean magnitude of relative error
MRE:	Magnitude of relative error
NABE:	Non-linear function adjusted analogy based estimation
PABE:	Probabilistic model of analogy based estimation
PRED(0.25):	Prediction at level 0.25
PSABE:	Project selection for analogy based estimation
RABE:	‘Regression toward the mean’ adjusted analogy based estimation
RBF:	Radial basis function networks
SABE:	Similarity function adjusted analogy based estimation
OLS:	Ordinary least square regression
SVR:	Support vector regression
SWR:	Stepwise regression

Chapter 1 Introduction

Recently, the software industry has faced a dramatic increase in the demand of new software products. On the other hand, software became more and more complex and difficult to produce and maintain. This demand-supply contradiction has contributed to the continuous improvements on software project management in which the ultimate goal is producing low cost and high quality software in short time. Successful software project management requires effective planning and scheduling supported by a group of activities, among which estimating the development cost (or effort) is fundamental to guide other activities. This task is known as *Software Cost Estimation*. Software cost estimation is a very active research field as it was more than 30 years ago, when the difficulties of estimation were discussed in "*The Mythical Man Month*" (Brooks 1975).

1.1 Software Cost Estimation

Cost estimation is a critical issue in project management (Chen 2007, Henry *et al.* 2007, Pollack-Johnson and Liberatore 2006). It is particularly important for software projects, as numerous software projects suffer from overruns (Standing 2004) and accurate cost estimation is one of the key points to the success of software project management.

Software cost (or effort) estimation is the process of predicting the amount of effort required to build a software system (Boehm 1981). It is a continuous activity which can or must start at the early stage of the software life cycle and continues throughout the life time. During the first phases of software life cycle, cost estimation is of necessity for software developing team to decide whether or not to proceed, though accurate estimates are obtained with great difficulties at this point due to the wrong assumptions or imprecise data. During the middle phases, the cost estimates are useful for rough validation and process monitoring. After completion, cost estimates are useful for project productivity assessment.

Since the software cost estimation affects almost all aspects of software project development such as bidding, budgeting, planning and risk analysis. The estimation has great impacts on software project management. If the estimation is too low, then the software development will be running under considerable constraints to finish the product in time, and the resulting software may not be fully functional or tested. On the other hand, if the estimation is too high, then too many resources will be committed to the project and this may result in significant amount of wasted resources. Furthermore, if the company is engaged in a contract, then too high an estimate may lead to loss of business opportunity.

Despite its importance, the estimation of software cost is still a weakness in software project management. Aiming at accurate and robust estimation, various cost estimation techniques have been proposed in past decades. Section 1.2 presents a brief introduction to these techniques including our research focus: analogy based estimation.

1.2 Introduction to Cost Estimation Methods

According to Angelis and Stamelos (2000)'s classification system, cost estimation methods can be grouped under three categories: expert judgment, algorithmic estimation, and analogy based estimation.

1.2.1 Expert Judgment Based Estimation

Expert judgment requires the consultation of one or more experts to derive the cost estimate (Hughes 1996). A Dutch study carried out by Heemstra (1992) revealed that 62% of estimators/organizations use this intuition technique and a study carried out later by Vigder and Kark (1994) also confirmed the widespread use of this technique. Despite its popularity this method seems to have received a poor reputation and it is often regarded as subjective and unstructured which makes it vulnerable compared with more structured methods (Angelis and Stamelos 2000).

1.2.2 Algorithmic Based Estimation

To date, the algorithmic method is the most popular technique in the literature. In algorithmic method, cost value is estimated by using certain mathematical function to link it to the inputs metrics such as ‘line of source code’ and ‘function points’. The mathematical model is often built upon some information abstracted from historical projects. Algorithmic method has some advantages over expert judgment: it has well defined formal structure; it produces identical outputs given the same inputs; it is efficient and good for sensitivity analysis (Selby and Boehm 2007).

The algorithmic method consists of a large number of techniques which can be further divided into two classes: function based methods and machine learning methods. Examples of function based methods are: COCOMO model (Boehm 1981), Function Points Analysis (Albrecht and Gaffney 1983), SLIM model (Putnam 1978), and Regressions (Schroeder et al. 1986). Examples of machine learning methods are: Artificial Neural Networks (Srinivasan and Fisher 1995), Classification and Regression Trees (CART) (Brieman et al. 1984).

1.2.3 Analogy Based Estimation

Analogy based estimation (Shepperd and Schofield 1997) is the process of identifying one or more historical projects that are similar to the project being developed and deriving the estimates from the similar historical projects. This technique is intended to mimic the process of an expert making decisions based on his/her experience. On the other hand, analogy based estimation has a concrete and well-defined estimation framework, given that similar past projects can be easily retrieved and the mechanism applying the nearest neighbors is correct. Thus, analogy based estimation is a very flexible method which allows the combination of the good aspects in both algorithmic methods and expert judgment. It has several advantages such as: it is able to deal with poorly understood domains, its output is relatively easy to interpret, and it offers the chance to learn from past experiences (Walkerden and Jeffery 1999).

1.3 Motivations

As explained in the previous section, analogy based estimation is one successful technique for cost estimation. However, it also has been criticized for relatively poor predictive accuracy, large computational expense, and intolerance to uncertainties. To overcome these drawbacks, many research works have been focusing on improving the four key components of analogy based system: similarity function, historical database, number of retrieved nearest neighbors and solution function (shown in Fig 1.1).

Similarity function (Shepperd and Schofield 1997), which measures the level of similarity between two different projects, is one of the key components in analogy based system. The choice of measure is an important issue since it affects the projects to be selected as the nearest neighbors. Many works (Auer et al., 2006, Huang and Chiu, 2006, Mendes et al., 2003) have been devoted to optimize the similarity function or feature weights, and the prediction accuracy of the analogy based system was reported to be significantly improved if the appropriate similarity functions or feature weights have been selected.

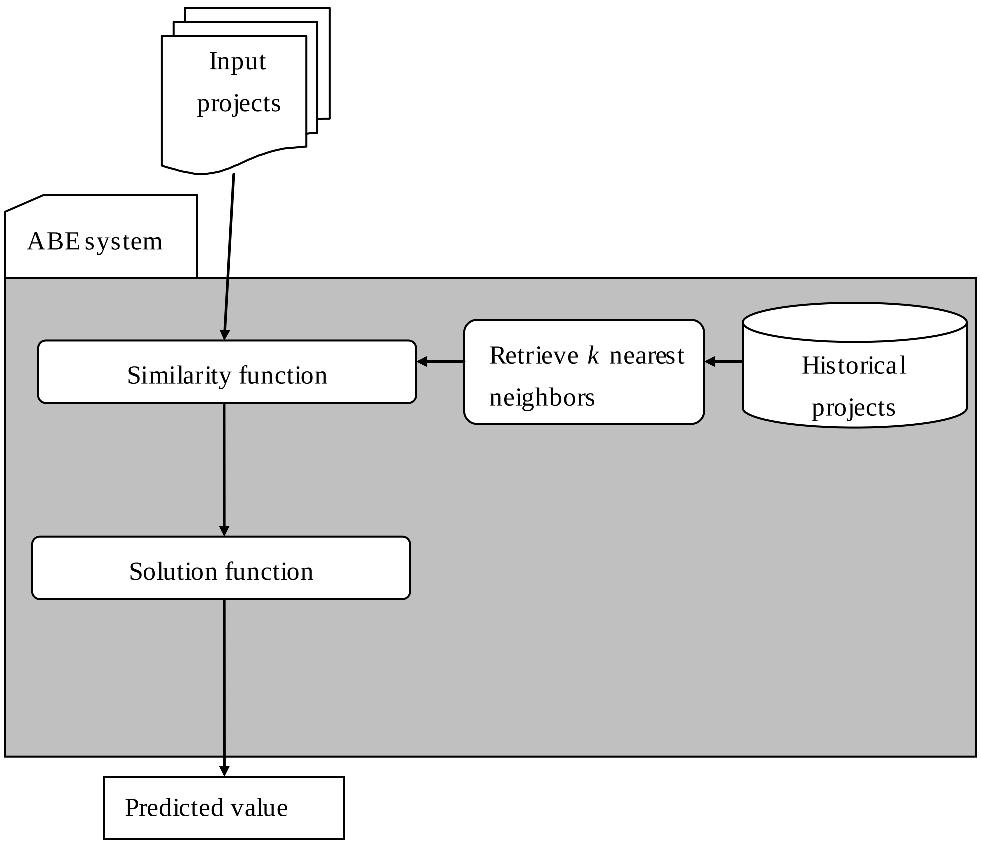

Figure 1.1: The ABE system structure

The historical database is the storage of the past projects' information, and it is used to retrieve the nearest neighbors. However, due to the instability of software development process the historical databases always contain noisy or redundant projects which might ultimately hinder the prediction accuracy of analogy based estimation. One possible solution is to reduce the whole database into smaller subset that consists of merely the representative projects.

Despite the importance of subset selection, very few research works (Kirsopp and Shepperd 2002) have been focused on this topic.

The number *K* of retrieved nearest neighbors decides how many nearest neighbors should be selected for the solution function to generate final prediction. Many works (Li and Ruhe. 2008, Mittas *et al.* 2008, Auer *et al.* 2006, Mendes *et al.* 2003, Leung 2002) have investigated the impacts of this value on the estimation results and/or considered optimizing this value. However, to our knowledge there is no widely accepted technique to choose *K* except the empirical trial-and-error method. Therefore, it is of great interest to develop systematic ways to optimize this parameter.

The solution function calculates the final estimation results from the nearest neighbors retrieved from the historical database. If an appropriate solution function is used, the prediction performance of analogy based system could be improved significantly. In the literature, only linear solution functions (Chiu and Huang, 2007, Jorgensen et al., 2003) have been considered though the relationships between the cost value and input features are usually non-linear. There is still a lack of research works to investigate the feasibility of applying non-linear solution functions.

As discussed above, many studies have been devoted to achieve accurate prediction by improving the four components of the analogy based system; however there still exists great opportunities to improve analogy based estimation for better performance. Moreover, most of the previous studies merely focused on improving accuracy which is one aspect of performance. The robustness, which is another important indicator, has received few concerns. As budget uncertainty is an important issue in project management (Yang 2005, Barraza and Bueno 2007), some authors pointed out that it is safer to generate probabilistic predictions such as probability distributions of the effort values or interval estimates with a probability. However, very little research (Angelis and Stamelos 2000, Jorgensen and Sjoberg 2003, van Koten and Gray 2006) has been done on probabilistic predictions.

1.4 Research Objective

The objective of this thesis is to improve accuracy, efficiency and robustness of analogy based estimation. Accuracy is the indicator of the cost estimator's ability to produce the quality predictions that match the software projects' costs. Efficiency is the speed of the cost estimator to complete a certain amount of estimation tasks. Robustness reflects the cost estimator's tolerance to uncertain inputs such as missing values and noisy data.

A number of journal/conference papers have been published under this objective. The research works that have been done are grouped into four chapters (each chapter is focused on one component of analogy based estimation): chapter 3 summarizes the works on mutual information based feature selection technique for similarity function; chapter 4 presents the research on genetic algorithm based project selection method for historical database; chapter 5 presents the work on non-linear adjustment to solution function; chapter 6 presents the probabilistic model of analogy based estimation which is focused on the number of nearest neighbors. The distribution of chapters 3 to 6 in the framework of analogy based system is illustrated in fig 1.2 where the shaded boxes with characters 'CH' stand for chapters (e.g. CH 3 stands for chapter 3). The remaining chapters in this thesis, namely chapters 2 and 7, are the literature review and the conclusions.

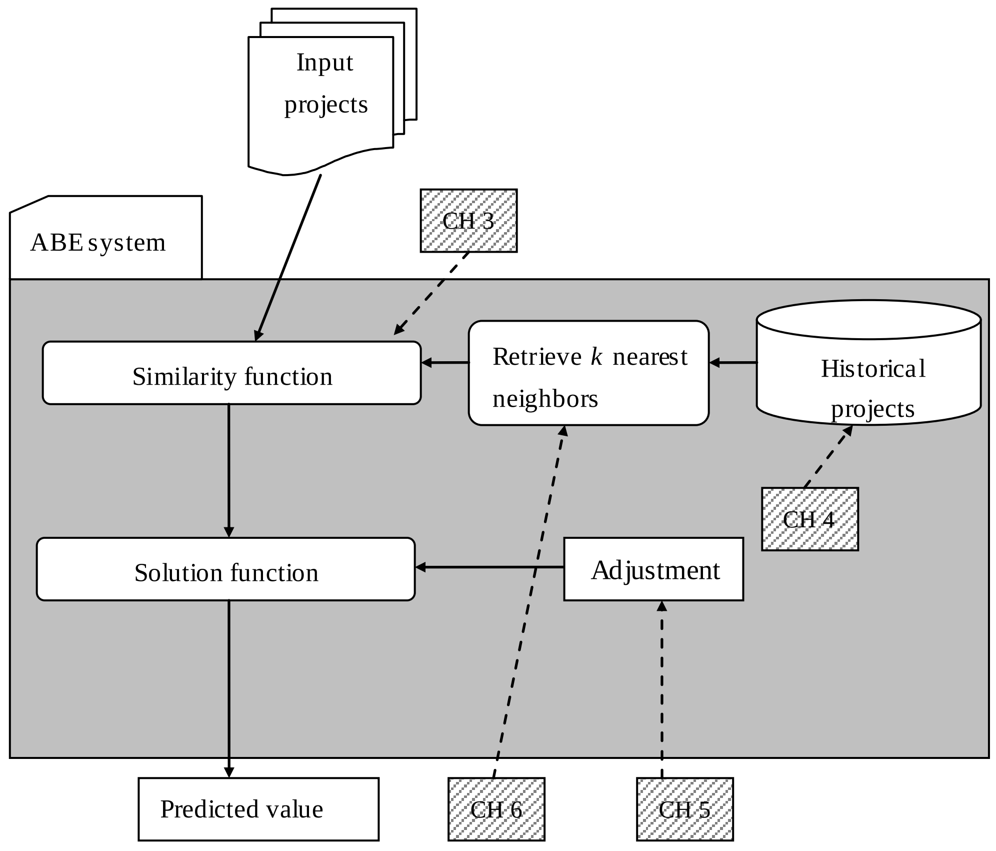

Figure 1.2: The distribution of research works

All of our research works share a common objective - enhance the analogy based estimation's capability to achieve more accurate results. In practice, this is very important for the software enterprises to maintain a better control of the budget throughout their software development processes. Theoretically speaking, these studies have contributed to the optimization of individual component of analogy based system. For instance, historical database and solution function have been largely refined or improved in our works. Furthermore, these studies point out a feasible direction to the global optimization of analogy based system.

Efficiency is another important aspect of estimation performance. In practice, improving estimation efficiency means enhancing the chance of winning bids. Many machine learning methods such as ANN and RBF can be very accurate in some situations, but they are often suffering from slow training speed. In addition, expert judgment could also be time consuming, as it usually takes time to gather/interview experts. Our studies on refining the historical dataset of analogy based system have achieved a significant reduction of unnecessary projects. Consequently, the efficiency of analogy based system is largely improved by our algorithm.

Moreover, the studies on probabilistic model lead to a more robust and reliable analogy based system. These studies could enhance the system's capability to deal with a broader scope of situations such as missing values and ambiguous inputs. Additionally, the probabilistic prediction provides a feasible way to model the inherited uncertainties and variabilities in the software development process.

As mentioned above, our research on analogy based estimation is of significant theoretical value and practical value. For a better understanding of our research work, the detailed background information of our research work is presented in the literature review in next chapter.

Chapter 2 Literature Review on Software Cost Estimation Methods

To obtain accurate software project cost estimates, various kinds of methods have been proposed. This chapter provides a detailed summary of the software cost estimation methods published in the past decade. The evaluation criteria for the prediction accuracy of these methods are also summarized and analyzed.

2.1 Introduction

In the literature there are several comprehensive overviews on the cost estimation methods, such as Walkerden and Jeffery (1997), Boehm *et al.* (2000), Briand and Wieczorek (2002), Jorgensen (2004a) and Jorgensen and Shepperd (2007). Among them, some reviews (Walkerden and Jeffery 1997, Boehm *et al.* 2000, Briand and Wieczorek 2002) have proposed different classification systems.

Walkerden and Jeffery (1997) introduced a system with four classes of estimation methods: empirical, analogical, theoretical, and heuristic. However, they stated that expert judgment cannot be included into their system. Moreover, there are overlaps between analogical and empirical, as analogical estimation process often involves empirical decisions (such as the choice of similarity measures in analogy based method) (Briand and Wieczorek 2002). Lately, Briand and Wieczorek (2002) defined a hierarchical scheme starting from two major classes (model-based methods, non-model-based methods) that are further divided into several sub-classes. The sub-classes contain further divisions and so on. Although the authors claimed that their system covers most types of estimation methods, the hierarchical system has a more complicated tree type structure with more intermediate nodes than other flatter systems and each intermediate node needs its own definition (such as 'data driven' and 'proprietary'). Boehm *et al.* (2000) proposed a simpler but comprehensive framework consisting of six major classes: parametric models, expert judgment, learning oriented techniques, regression based methods, dynamic based models, and composite methods. Directly under each major class are the estimation methods and this system can include most types of estimation methods (Boehm *et al.* 2000). Our classification system is modified from Boehm's framework with the consideration to balance the number of recent publications under each major class.

2.2 Literature Survey and Classification System

Prior to our classification system, a structured literature survey is conducted to select the related journal papers during the period between 1999 and 2008. The keywords used for searches in SCI engine are 'software cost estimation', 'software effort estimation', 'software resource estimation', 'software effort prediction', 'software cost prediction', 'software resource prediction', and 'software prediction'. The main criterion for including a journal paper in the survey is that the paper presents research on software development effort or cost estimation. Papers related to prediction of software size/defects, modeling of software process, or identification of factors correlated with software project cost, are included only if the main purpose of the study is to improve software cost estimation. The papers with pure discussions or opinions are excluded. The process above results in a collection of 158 journal papers.

To construct our classification system, we first calculate the number of publications under each category in Boehm (2000)'s system. The results reveal that the recent research trend has different emphases on each category, for example there are more than 80 papers related to 'learning oriented techniques' while only 5 papers and 4 papers under 'dynamic based models' and 'composite methods' respectively. In addition, Boehm's scheme does not include the discrete event simulation model which has only recently appeared as one promising technique. Moreover, there are 35 papers related to 'analogy based estimation' which stands for the largest proportion among the 'learning oriented techniques'.

For a more balanced structure, we combine the classes 'dynamic based models', 'composite methods' and other emerging methods (such as discrete event simulation) to form the category ‘Other methods’. Furthermore, we split the ‘analogy based estimation’ from the ‘learning oriented techniques’ to be a major class, and we rename the remaining methods under ‘learning oriented techniques’ as ‘machine learning techniques’. The reason for this splitting is that analogy based method is the learning oriented method with highest amount of publications and many previous studies (Walkerden and Jeffery 1997, Angelis and Stamelos 2000) have already regarded it as one major class. Analogy based estimation is particularly popular in the context of software cost estimation which might be due to the fact that analogy based estimation build up the connections between project managers making cost estimation based on the memories of past experiences and the formal use of analogies in Case Based Reasoning (CBR) (Kolodner 1993).

From the discussion above, our classification system is established in Fig 2.1. It contains six major categories: expert judgment, parametric models, regressions, machine learning methods, analogy based estimation, and other methods.

Based on our classification system, the number of publications per year of each major class is summarized in table 2.1. It is seen that regressions and machine learning methods are the most popular methods in the past decade. Parametric models and analogy based estimation rank at the third place.

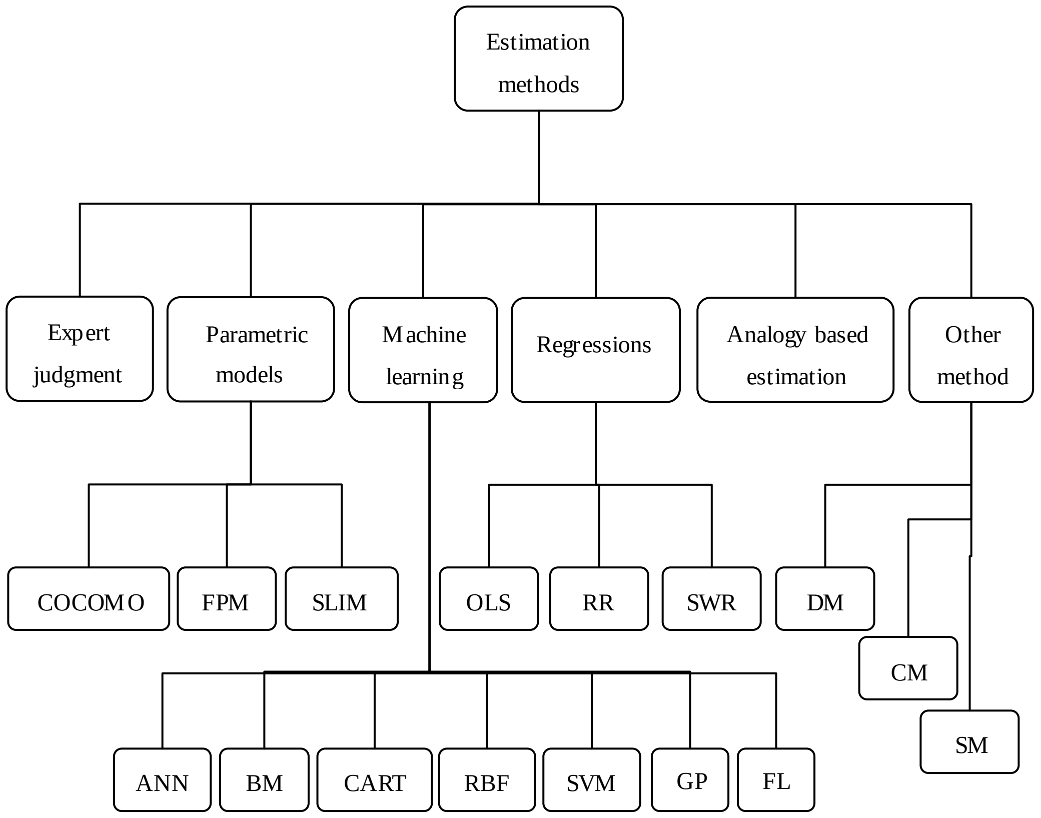

Figure 2.1: The classification of software cost estimation methods

COCOMO: constructive cost model, FPM: Function point model, SLIM: software life-cycle model, ANN: artificial neural networks, BM: Bayesian methods, CART: classification and regression trees, RBF: radial basis functions, SVM: support vector machine, GP: genetic programming, FL: fuzzy logic, OLS: ordinary least-square regression, RR: robust regression, SWR: stepwise regression, DM: dynamics models, CM: composite methods, SM: simulation models.

Table 2.1: Number of publications in each year from 1999 to 2008

Year	EJ	PM	RE	ML	AB	OT
1999	2	3	4	1	1	1
2000	1	2	5	5	3	1
2001	3	4	8	6	5	3
2002	0	4	4	4	1	1
2003	4	2	6	5	6	2
2004	7	1	3	3	1	2
2005	3	3	6	6	1	1
2006	2	6	5	8	3	4
2007	3	6	8	5	5	3
2008	3	4	10	10	9	2
Total	*28*	*35*	*59*	*53*	*35*	*20*

EJ: expert judgment, PM: parametric models, RE: regressions

ML: machine learning methods, AB: analogy based estimation, OT: other methods

To investigate the trends of publications, the proportion of each class from 1999 to 2008 is depicted in the bar-charts of fig 2.2. The whole period is divided into three nearly equal segments: 1999 – 2001, 2002 – 2004, and 2005 – 2008. Fig 2.2 suggests that:

- Regression technique is the most frequently used method. This observation confirms with Jorgensen and Shepperd (2007)'s survey. Among the regression papers, a large number of papers use regressions to compare with the estimation methods they propose.

- The proportion of papers on machine learning methods is constantly increasing and they have the same proportion of publications as regressions have in recent 4 years. Unlike regression papers, majority of machine learning papers introduce or propose new cost estimation techniques.

- The proportions of papers on parametric models and analogy based estimation are around 15% with some small fluctuations.

- The popularity of expert judgment based estimation was at its highest in the period 2002-2004.

- The proportion of 'other methods' is around 8% throughout the past decade.

- The distributions of the papers become more and more even, as in the period after 2001 no method stands for a proportion larger than 25%. This observation is one supportive evidence for our modifications to Boehm's classification system.

In the following sections, a comprehensive review is presented for each major class.

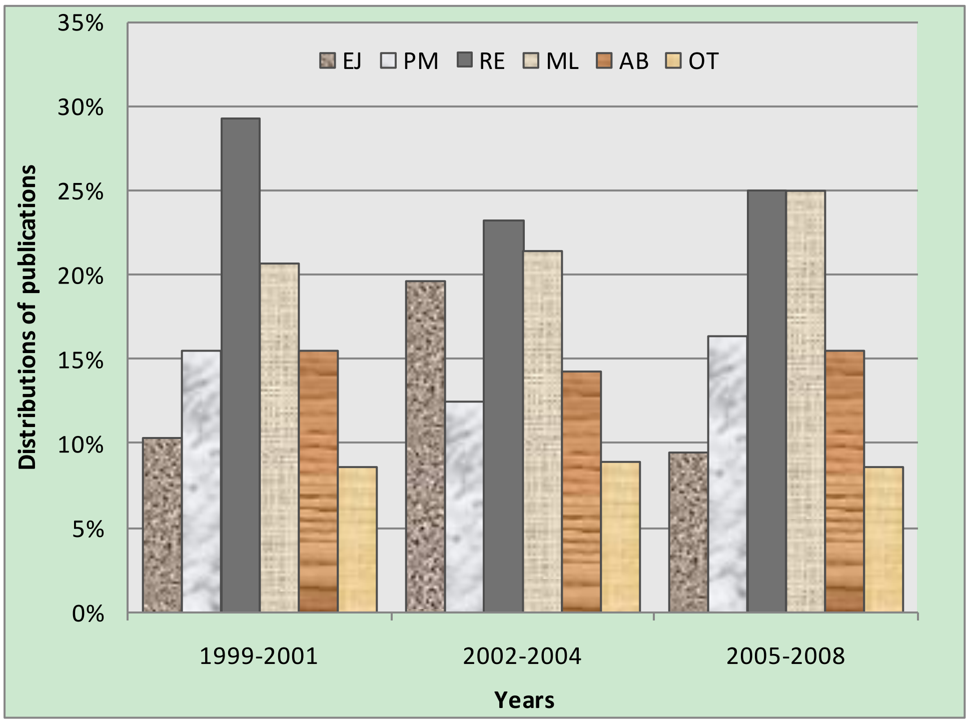

Figure 2.2: The distribution of publications of each class during 1999 - 2008

EJ: expert judgment, PM: parametric models, RE: regressions, ML: machine learning methods, AB: analogy based estimation, OT: other methods

2.3 Cost Estimation Methods

2.3.1 Expert Judgment

Expert judgment requires the consultation of one or more experts to derive the cost estimate (Hughes 1996). With their experience and understanding of the new project and the experience from past projects, the experts could obtain the estimation by a non-explicit and non-recoverable reasoning process, i.e., "intuition". As reported in the business forecasting study conducted by Blattberg and Hoch (1990), most estimation processes have both intuitive and explicit reasoning elements. In fact, even formal software cost estimation models may require expert estimates as important input parameters (Pengelly, 1995). Jorgensen (2004a) presented an extensive review of studies related to the expert estimations conducted before 2003. As a subsequent work of Jorgensen (2004a)'s, we focus on the expert judgment studies published after 2003. Expert judgment often encounters a number of issues, such as estimate uncertainty, bias caused by over-optimism, and etc. A number of research works are aiming to solve these problems.

To describe the uncertainty of cost estimate, Jorgensen and Sjoberg (2003) proposed and evaluated a Prediction Interval (PI) approach, which is based on the assumption that the estimation accuracy of earlier software project predicts the cost PIs of new projects. Lately, Jorgensen et al. (2004) conducted four studies on expert judgment based PIs. The results suggest that the PIs were generally much too narrow to reflect the chosen level of confidence. Moreover, Jorgensen (2004b) claimed that the traditional request for PIs is not optimal and leads to overoptimistic views about the level of estimation uncertainty.

Many works are devoted to the study of the over-optimism phenomenon. Moløkken and Jørgensen (2005) observed that people with technical competence provided more overoptimistic estimates than those with less technical competence. Jørgensen et al. (2006) examined the degree to which level of optimism in software engineers' predictions is related to optimism on previous predictions. Jørgensen et al. (2007) concluded that optimistic software engineers have a number of characteristics such as higher confidence in their own predictions, lower development skills, poorer ability or willingness to recall effort on previous tasks, and etc. Some techniques are proposed to reduce the bias towards over-optimism. Jorgensen (2005) provided some evidence based guidelines for assessing the uncertainties in expert judgment. Moløkken and Jørgensen (2004) propose an approach combining the judgments of experts with different backgrounds by means of group discussion.

In addition, other studies summarize different characteristics of expert judgment. Jorgensen and Sjoberg (2004) discovery that customer expectations of a project's total cost can have a very large impact on expert judgment. McDonald (2005) shows that cost estimates are dependent upon two kinds of team experience: (1) the average experience for the members of each team and (2) whether or not any members of the team have similar project experience. Grimstad and Jørgensen (2007) reported a high degree of inconsistency in the previous experts' estimates. Jorgensen (2004d) suggested that the recall of very similar previously completed projects seemed to be a pre-condition for accurate top-down based estimates.

Although expert judgment has been used widely, the estimates are obtained in a way that is not explicit and consequently difficult to be repeated. Nevertheless, expert judgment can be an effective estimate tool when used as an adjustment factor for algorithmic models (Gray *et al.* 1999).

2.3.2 Parametric Models

Parametric models are defined by mathematical formula and need to be calibrated to local circumstances in order to establish the relationship between the cost and one or more project features (cost drivers). Usually, the principal cost driver used in such models is software size (for instance, lines of source code, the number of function points, pages, etc.). This section includes three function methods, COCOMO (Boehm, 1981), Function Points Analysis (Albrecht and Gaffney, 1983), and SLIM model (Putnam, 1978).

COCOMO (Constructive Cost Model)

COCOMO I is one of the best known and best documented software cost estimation model (Boehm 1981). It is a set of three modeling levels: basic, intermediate, and detailed. The basic COCOMO takes the following relationship between cost (effort) and size:

$$Y = a(KLOC)^b \tag{2.1}$$

where Y is the project effort/cost, $KLOC$ represents the size in terms of thousands of lines of source code, and the coefficients a and b depend on COCOMO's modeling level and the mode of the project to be estimated (organic, semidetached, embedded). In all cases, the value of b is greater than 1. The intermediate and detailed COCOMO takes the following general form:

$$Y = a(KLOC)^b \prod_{i=1}^{15} EM_i \tag{2.2}$$

where EM$_i$ is the *i*th effort multiplier. Effort multiplier is the parameter that affects effort the same degree regardless of project size. However, COCOMO together with its Ada (Kaplan 1991) update are prone to difficulties in estimating the costs of software developed in new lifecycle processes and capabilities (such as iterative model and spiral model).

The research on COCOMO II started in 1994. COCOMO II (Boehm *et al.* 1995) has two models (early design and post architecture) for cost estimation at different development stages. Early design model is used in the initial stages of a software project when very little information is known about the product being developed. The post architecture model is the most detailed estimation model and it is used when software lifecycle architecture has been developed. The early design and post architecture models share a common form:

$$\begin{aligned} Y &= a(KLOC)^b \prod_{i=1}^{n} EM_i \\ b &= 1.01 + 0.01 \sum_{j=1}^{5} (scale\ \ factor)_j \end{aligned} \tag{2.3}$$

where the five 'scale factors' are the parameters that have large influence on big projects and small influence on small projects (which is different from the effort multipliers). The scale factors are precedentedness, development flexibility, risk resolution, team cohesion, and process maturity. Early design model and post architecture model have different number (n) of effort multipliers. Detailed descriptions about the effort multipliers can be found in (Boehm *et al.* 1995)

Lately, a lot of research works have been done on the COCOMO models. Chulani *et al.* (1998) proposed a new version of COCOMO II model which includes a 10% weighted average approach to adjust prior expert determined model parameters. Moreover, Chulani *et al.* (1999) introduced the Bayesian inference for the tuning of the expert determined model parameters. Jongmoon *et al.* (2002) proposed a way of integrating CASE tool into COCOMO II and their approach resulted in an increase in the prediction accuracy. Benediktsson *et al.* (2003) introduced the COCOMO-style cost model for the incremental development and explore the relationship between effort and the number of increments. Han *et al.* (2005) adopted COCOMO model for software project financial budget optimization. Huang *et al.* (2007) proposed a novel neuro-fuzzy COCOMO model and the authors report that this model greatly improves estimation accuracy. More recently, Fairley (2007) provided a comprehensive overview on COCOMO models. This paper presents a summary of recent work on COCOMO modeling and provides future directions for COCOMO-based education and training.

Function Points Model (*FPM*)

The function point (FP) measure was first developed by Albrecht (1979) as an alternative to lines of code for measuring the software size. The function point method defines five basic function types to estimate the size of the software. The five functions types are internal logical files (ILF), external interface files (EIF), external inputs (EI), external outputs (EO), and external inquiries (EQ).

Based on the definition of function points, a number of researchers (Albrecht and Gaffney 1983, Kemerer 1987, Matson *et al.* 1994, Abran and Robillard 1996) used FP for cost estimation. In their studies, each function point is first classified into one of three complexity levels: low, average or high. Then an integer complexity value is assigned to the function point based on the ordinal scale complexity classification. Furthermore all the identified function complexity values are added together to derive an unadjusted function point count (FPC). Additionally, this count is often adjusted by up to 14 technical complexity factors that account for a variety of non-functional system requirements (e.g. performance, reliability, backup and recovery etc.) to give an adjusted function point count (AFPC). The resulting counts are then used to derive the cost estimate by using the following form:

$$Y = a + b \times FPC(AFPC) \tag{2.4}$$

where a and b are the coefficients determined by ordinary linear regression method. As the software industry keeps evolving rapidly, many other types of size metrics are developed, such as Weighted Methods per Class (WMC), Number Of Children (NOC) (Chidamber and Kemerer 1994), and Class Point (CP) (Costagliola et al. 2005). However, many current papers still considered function point as one of the critical factors in their cost models (Kitchenham et al. 2002, Ahn et al 2003, Moses and Farrow 2005).

Software Life-cycle Model (SLIM)

Putnam (1992) first developed the Software Life-cycle Model (SLIM). The basic assumption of SLIM is that the Rayleigh distribution (See Fig 2.3) can be used to model the change of staff levels on large software projects which have more than 70,000 'Thousands of Delivered Source Instruction's (KDSI). It is assumed that the number of people working on a project is a function of time. A project starts with relatively few people and the manpower reaches a peak and then falls off. The decrease in manpower during the testing is less than that during the earlier construction phase. In addition, Putnam explicitly excluded requirements analysis and feasibility studies from the life cycle.

The basic Rayleigh curve (Fig 2.3) defining the effort distribution is described by the following differential equation:

$$\frac{dy}{dt} = 2Kat\exp(-at^2) \tag{2.5}$$

where t is elapsed time from the starting point of a software project, K is the total project effort, and a is a constant that determines the shape of the curve.

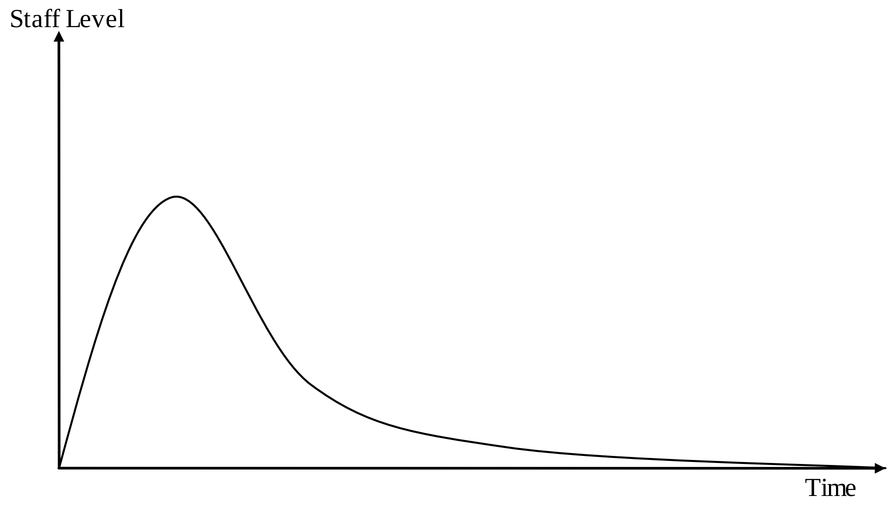

Figure 2.3: Rayleigh function in SLIM model

In order to obtain the total project effort K and development time t_d, the following two formulas can be derived after a few algebraic manipulations:

$$\begin{aligned} t_d &= \left[\frac{S^3}{D_0 C^3}\right]^{\frac{1}{7}} \\ K &= \left(\frac{S}{C}\right)^{\frac{9}{7}} D_0^{\frac{4}{7}} \end{aligned} \tag{2.6}$$

where S is the system size measured by KDSI (Thousands of Delivered Source Instructions), D_0 is the manpower acceleration, and C is the technology factor. SLIM does not gain much popularity as COCOMO and FPM. However, in the early 2000's the company named 'Quantitative Software Management' has developed a successful package of three tools based on Putnam's SLIM. These include SLIM-Estimate, SLIM-Control and SLIMMetrics. SLIM-Estimate is a project planning tool, SLIM-Control is a project tracking and oversight tool, and SLIM-Metrics is a software metrics repository and benchmarking tool. More information on these SLIM tools can be found at http://www.qsm.com.

2.3.3 Regressions

According to our survey, regression methods are most popular in the past decade. The most commonly used regressions method is the Ordinary Least Square (OLS) regression which has also been criticized for its restrictive assumptions and poor performance. This section also includes other types of regression such as robust regression and stepwise regression. These techniques are regarded as the improved version of OLS regression.

Ordinary least-square regression (OLS regression)

OLS regression is one of the most commonly used models for cost estimation. In general, a linear regression has the following form:

$$\hat{Y} = a + b_1 X_1 + b_2 X_2 + \ldots + b_n X_n + e \qquad (2.7)$$

where $\hat{Y}$ denotes the dependent variable (project cost/effort), X_i stands for independent variables (project features/cost drivers), and b_i is the so called regression coefficient, a is referred as the intercept, and the error term e is a random noise with a normal distribution.

The OLS regression has a number of strong assumptions. One important assumption is the so called homoscedasticity which means that the differences between the actual values and the predicted values do not change under different values of X_i. Another assumption is that OLS variables are all continuous in nature. Thirdly, OLS regression requires that there are no outlier values in both independent and dependent variables. However, extreme outliers are commonly found in software engineering dataset, probably due to the misunderstandings or lack of precision in the data collection process. Finally, no missing data is allowed in OLS regression. On the contrary, missing data is often reported when there is limited time and budget for data collection. In all, many of the difficulties discussed above can be solved by some advanced techniques such as robust regression, logistic regression and data imputation. However these advanced techniques remain difficult to be implemented by most engineers and managers, and applying them still requires extensive training and experience (Briand and Wieczorek 2002).

Although OLS regression is one of the oldest methods for cost estimation, it is still widely applied and continuously improved for more accurate predictions. Kitchenham (1998) proposed analysis of variance (ANOVA) and OLS regression to analyze unbalanced data sets. Angelis *et al.* (2001) proposed categorical regression (CATREG) for the datasets with large number of categorical attributes, such as ISBSG (ISBSG, 2007) dataset. Sentas *et al.* (2005) modified the standard OLS regression to produce the interval predictions. Jeffery *et al.* (2000) applied OLS regression on both ISBSG data and company specific data with comparison against analogy based method. More recently, Jorgensen (2004) conducted some regression analysis of cost estimation on a data collection of 49 software development projects. Lucia *et al.* (2005) applied multivariate OLS regression for corrective maintenance effort estimation. Mendes *et al.* (2005) applied multivariate OLS regression for Web effort estimation. Multivariate OLS has identified 'total number of Web pages' and 'features provided by the application' to be the two most influential effort predictors. Costagliola *et al.* (2005) applied multivariate OLS regression to predict development effort of object oriented systems by using class points.

Robust Regressions (RR)

Robust regressions are an improved version of OLS regression. They alleviate OLS regression's sensitivity to outliers. Instead of minimizing the sum of square of absolute error in OLS regression, robust regressions use other objectives for optimization. There are several types robust regression such as LMS (least median of squares) which minimizes the median of square of absolute error (Rousseuw and Leroy, 1987), LBRS (least-squares of balanced relative errors) which minimizes the sum of squares of balanced relative error, and LIBRE (least-squares of inverted balanced relative errors) which minimizes the sum of squares of inverted balanced relative error (Miyazaki *et al.* 1994).

Another approach that can be regarded as robust regression is a technique that only uses the data points lying within two (or three) standard deviations of the mean response variable (Boehm *et al.* 2000). This method automatically filters out outliers and it can be used only when there are sufficient observations. Although this technique has the weakness of eliminating outliers without direct reasoning, it is still very useful for developing software estimation models on the dataset where there are only a few project features.

Stepwise regression (SWR)

Stepwise regression (Schroeder *et al.* 1986) is based on an important assumption that some independent variables in a multivariate regression do not have an important explanatory effect on the dependent variable. If this assumption is true, to keep only the statistically significant variables is a convenient simplification. Usually, stepwise procedure takes the form of a sequence of *F*-tests, but other techniques are also applicable, such as *t*-tests and adjusted *R*-square. The stepwise regression main approaches are: (1)

forward selection, which involves starting with no variables in the model, trying out the variables one by one and including them if they are statistically significant; and (2) backward selection, which involves starting with all candidate variables and testing them one by one for statistical significance, deleting any that are not significant. Stepwise regression has been frequently employed for cost estimation (Shepperd *et al.* 1997, Shepperd and Kadoda 2001, Mendes *et al.*, 2003).

2.3.4 Machine Learning

Machine learning (ML) methods imitate some functionality of human mind and allow us to deal with large and complex problems at a relatively high speed (Schank 1982). The ML techniques have been successfully applied to many difficult problems such as pattern recognition, biology, stock market analysis, and etc. Recently they become increasingly popular in software cost estimation research. In literature, Classification and Regression Trees (Brieman *et al.* 1984), Bayesian Methods (Chulani *et al.* 1998), and Artificial Neural Networks (Lawrence, 1994) are the most common ML techniques. Other ML techniques (such as radial basis function, support vector machine, and genetic programming) are also introduced for cost estimations. This section provides a detailed overview on ML methods.

Classification and Regression Trees (CART)

The classification and regression tree method was first proposed by Brieman *et al.* (1984). This method is originally a non-parametric and tree structured analysis procedure that can be used for classification. Lately the trees are used for problems with numerical targets, so they are named as regression trees. Being the combination of both types of trees, the total method is called classification and regression tree (CART).

The construction of the CART involves recursively splitting the data set into (normally two) relatively homogeneous subsets until the terminate conditions (for numerical variables e.g. Q: is weight > 50? And for categorical variables e.g. Q: is transparency high?) are satisfied. The partition is determined by splitting rules associated with each of the internal nodes. Each instance in the data set is assigned to a unique leaf node, where the conditional distribution of the response variable is determined. The best tree is determined by cross-validation using a spread minimization criterion.

CART provides additional information about the tree generated. At each partition, it gives a list of 'competition' and 'surrogates' for the independent variables. The variables with 'competition' tag will be kept for the next split. 'Surrogate' variables are highly correlated with the independent variables used to partition the data and surrogate variables could be used as alternative factors.

CART has the following advantages: the capability of dealing with categorical features, the easily understandable diagram of complex data and the ability to identify the major subsets in the total dataset (Srinivasan and Fisher 1995). Due to these advantages, CART is frequently adopted by researchers in cost estimation area (Briand *et al.*1998, Kitchenham 1998, Briand *et al.* 1999, Khoshgoftaar *et al.* 1999, Pickard *et al.* 1999, Stensrud .2001, Stewart 2002, Mendes *et al.* 2003).

Bayesian Methods

Chulani *et al.* (1999) criticized the traditional software effort estimation models that software engineering data sets do not follow the parametric assumptions and traditional models do not provide any support for risk assessment and mitigation. They first proposed the Bayesian inferences to address these problems. Bayesian inference provides posterior distributions for model parameters of interest by the following formula:

$$f(\theta \mid X) = \frac{f(X \mid \theta) f(\theta)}{f(X)} \tag{2.8}$$

where $f(\theta \mid X)$ is the posterior distribution of the parameter θ given the distribution of the data sample *X*, $f(X)$ is the distribution of data sample *X*, $f(\theta)$ is the prior distribution of parameter θ, which represents knowledge about the parameter prior to data collection (Gelman *et al.*, 1998), and $f(X \mid \theta)$ is the sampling distribution representing the distribution of the data sample *X* given the parameters used to model the data.

Since Bayesian inference is a promising technique to integrate information from different sources, it gains significant popularity in software cost estimation. For example, Jongmoon *et al.* (2002) employed Bayesian inference to combine two sources of information, from expert-judged and data-determined, to increase prediction accuracy. Many other recent studies also use Bayesian inference, such as Moses (2002), Moses and Farrow (2003), Moses and Farrow (2005), Van Koten and Gray (2006).

Besides Bayesian inference, the Bayesian Belief Networks (BBN) also receives increasing concerns as a successful alternative for uncertainty modeling. The main concepts behind Bayesian inference also hold for BBN. The BBN is a directed acyclic graph describing probabilistic cause-effect relations among the linked nodes. Each node represents a random variable that can takes discrete or continuous values according to a probability distribution, which can be different for each node. Each influence relationship is represented by an arc starting from the influencing variable (parent node) and ending on the influenced variable (child node). The independence (conditional) of two variables can be determined by the conditions of d-separations (Pearl 1988).

BBN is adopted by many authors for cost estimation. Stewart *et al.* (2002) investigated the utility of the Naive-Bayes classifier which is a special kind of BBN. Stamelos *et al.* (2003) illustrated the use of BBN to support expert judgment for software cost estimation. Pendharkar *et al.* (2005) illustrated how a belief updating procedure can be used to incorporate decision-making risks.

Artificial Neural Network (ANN)

Artificial neural network (ANN) is one of the machine learning techniques that have played an important role in solving complex problems with difficult or unknown analytical solution (Lawrence, 1994). It has become an important element in approximating nonlinear relationships.

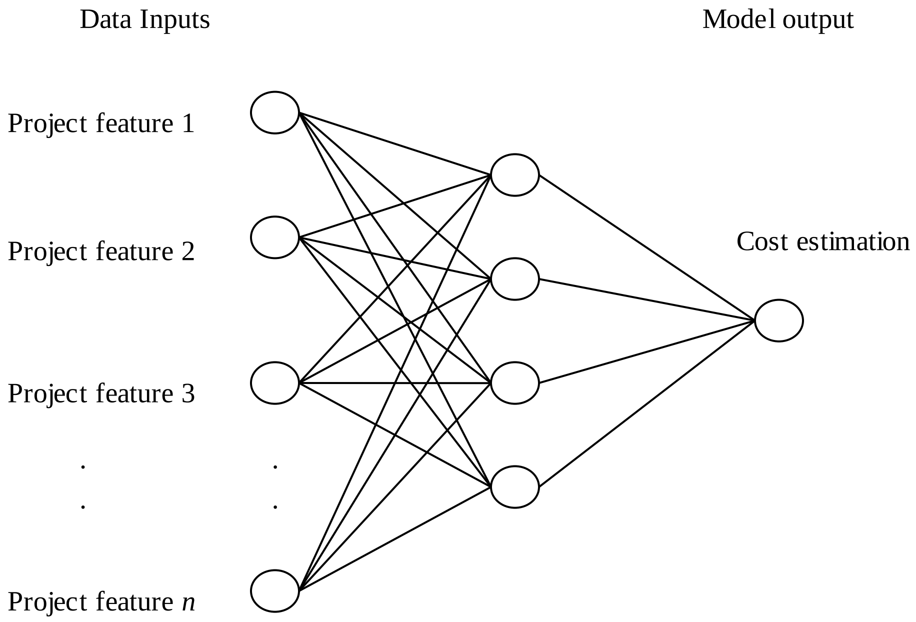

Figure 2.4: An example of artificial neural network

The inputs and outputs are linked according to specific topologies where each neuron is connected to at least one other neuron in a mesh-like fashion.

There are three distinct layers in a neural network: the input layer, the hidden layer(s), and the output layer. The connections of neurons across layers represent the transmission of information between neurons. Fig 2.4 depicts a three layer network consisting of a stream of input project features to the input layer, a hidden layer of some neurons and an output layer with cost estimate as the output value.

Due to its good approximation capability, neural network has been frequent studied/applied for cost estimation. Many studies aim to improve the performance of ANN. Srinivasan and Fisher (1995) first proposed neural network for cost estimation. Samson *et al.* (1997) introduced the Albus perceptron based neural network for cost estimation. Lee *et al.* (1998) integrated neural network with cluster analysis. Shukla (2000) proposed a neural network (NN) predictor trained genetic algorithm. Eung and Jae (2001) proposed a search method that finds the right level of relevant cases for the neural network model. Other studies simply adopted NN as a candidate method for the comparisons against their estimation methods (Finnie et al.1997, Gray and MacDonell 1997, Wittig and Finnie 1997, Gray and MacDonell 1999, Burgess and Lefley 2001, Shepperd and Kadoda 2001, Heiat 2002, Pendharkar and Subramanian 2002, Mair *et al.* 2000, Heiat 2002, de Barcelos et al. 2007).

Other Machine Learning Methods

In addition to the techniques described above, many different types of machine learning methods also appeared in the literature. Examples are Radial Basis Function (Shin and Goel 2000, Dri *et al.* 2006), Support Vector Machine (Vapnik 1995, Adriano 2006), Genetic Algorithm/Programming (Shukla 2000, Burgess and Lefley 2001, Aguilar-Ruiz *et al.* 2001), and Fuzzy Logic (Ahmeda *et al.* 2005, Engel and Last 2007).

2.3.5 Analogy Based Estimation

Analogy based estimation (ABE), which was first proposed by Sternberg (1977), is essentially a case-based reasoning (CBR) approach (Shepperd and Schofield 1997). The principle of ABE is relatively simple: when provided a new project, it identifies one or more historical projects that are similar to the current project and then derives the final estimates from these nearest neighbors. Generally, ABE consists of four components: similarity function, historical database, number of retrieved nearest neighbors and solution function (See Fig 1.1). The ABE system procedure normally consists of the following four stages:

- Collect the past projects' information and prepare the historical data set
- Select current project's features such as Function Points (FP) and Lines of Source Code (LOC), which are also collected with past projects
- Calculate the similarities between new project and the past projects, and

identify the nearest neighbors. The commonly used similarity function is the reciprocal of weighted Euclidean distance.

- Predict the cost of the new project from the chosen nearest neighbors by using the solution function. Usually the mean value function is used as solution function.

Aiming to improve ABE's performance, many works have been devoted to improve its four components. The following paragraphs present detailed descriptions of these components and summarize published works under these components:

Similarity Function

The similarity function, which measures the level of similarity between two different projects, is one of the key components in ABE. The choice of measure is important since it affects which projects are selected as the nearest neighbors. The similarity function has the general form (Li et al. 2007):

$$Sim(p, p') = f\left(Lsim(f_1, f_1'), Lsim(f_2, f_2'), \ldots, Lsim(f_n, f_n')\right) \qquad (2.9)$$

where p and p' denote any two projects, f_i and f_i' denote the features of project, n is the number of features in each project, and $Lsim(\bullet)$ is the so called local similarity function of every project feature. The function $f(\bullet)$ and $Lsim(\bullet)$ together define the structure of similarity function. All types of similarity functions are special cases of this general form. Among various types of similarity functions, Euclidean distance based similarity (ES) and Manhattan distance based similarity (MS) are most popular. The Euclidean similarity is based on the Euclidean distance between two projects:

$$Sim(p, p') = 1 \Big/ \left[\sqrt{\sum_{i=1}^{n} w_i Dis(f_i, f'_i)} + \delta \right]$$

$$Dis(f_i, f'_i) = \begin{cases} (f_i - f_i')^2, & \text{if } f_i \text{ and } f_i' \text{ are numeric or ordinal} \\ 1, & \text{if } f_i \text{ and } f_i' \text{ are nominal and } f_i = f_i' \\ 0, & \text{if } f_i \text{ and } f_i' \text{ are nominal and } f_i \neq f_i' \end{cases}$$

(2.10)

where *p* and *p'* denote any two projects, f_i and f_i' denote the *i*th features of projects *p* and *p'* respectively, $w_i \in [0, 1]$ is weight of *i*th feature, δ is a small constant to prevent the situation that the denominator equals 0, and *n* is the total number of features. The Manhattan similarity is based on the Manhattan distance which is the sum of the absolute distances for each pair of features.

$$Sim(p, p') = 1 \Big/ \left[\sum_{i=1}^{n} w_i Dis(f_i, f'_i) + \delta \right]$$

$$Dis(f_i, f'_i) = \begin{cases} |f_i - f_i'|, & \text{If } f_i \text{ and } f_i' \text{ are numeric or ordinal} \\ 1, & \text{If } f_i \text{ and } f_i' \text{ are nominal and } f_i = f_i' \\ 0, & \text{If } f_i \text{ and } f_i' \text{ are nominal and } f_i \neq f_i' \end{cases}$$

(2.11)

In the literature, there are many other types of similarity measures, such as Maximum distance based similarity (Angelis and Stamelos 2000), Minkowski distance based similarity (Angelis and Stamelos 2000), and rank mean similarity (which is the mean value of the ranking of each project feature) (Walkerden and Jeffery 1997). A summary of the similarity functions used in a previous paper is provided in table 2.2. Table 2.2 shows that Euclidean similarity is the most popular similarity function, as it has the straightforward geometrical definition of the distance between two points in the k-dimension Euclidean space.

Table 2.2: Summary of different similarity functions

Source	*Euclidean Similarity*	*Manhattan Similarity*	*Maximum Similarity*	*Minkowski Similarity*	*Rank Mean*
Shepperd and Schofield (1997)	Yes	No	No	No	No
Walkerden and Jeffery (1999)	No	No	No	No	Yes
Angelis and Stamelos (2000)	Yes	Yes	Yes	No	No
Leung (2002)	No	Yes	No	No	No
Mendes et al. (2003)	Yes	No	Yes	No	No
Jorgensen et al. (2003)	Yes	No	No	No	No
Auer et al. (2006)	Yes	No	No	No	No
Huang and Chiu (2006)	Yes	No	No	No	No
Chiu and Huang (2007)	Yes	Yes	No	Yes	No
Li et al. (2007)	Yes	No	No	No	No
Mittas et al. (2008)	Yes	No	No	No	No
Li and Ruhe (2008a)	Yes	No	No	No	No
Li and Ruhe (2008b)	Yes	No	No	No	No
Keung et al. (2008)	Yes	No	No	No	No
Totals	12	3	2	1	1

In the literature, there are some works comparing the performances of different similarity functions. Angelis and Stamelos (2000) concluded that the Euclidean similarity, Manhattan similarity, and Maximum similarity produced almost the same results. However, they also state that this result may be affected by the choices of data set. Mendes et al. (2003) have found out that the maximum similarity for one and three nearest neighbors produces statistically worse results than Euclidean similarity. Chiu and Huang (2007) claimed that the differences among Euclidean similarity, Manhattan similarity and Minkowski similarity are trivial. In all, there is still no solution to the problem of under which condition what type of similarity function is preferable. The relationship between choice of similarity function and characteristics of historical dataset is worth explorating.

Using the similarity functions in (2.10) and (2.11), it is reasonable to conjecture that different features may have different importance to the total similarity (for example, in many cost models the feature 'function point' is more important than the feature 'programming language'). Moreover, many researchers point out that there exist large potentials to improve ABE's accuracy by assigning appropriate weights to the right feature. In this direction, several research works are focusing on determining the optimal weight of each feature (feature weighting). Shepperd and Schofield (1997) set each weight to be either 1 or 0 and then apply a brute-force approach to choose optimal weight values, Walkerden and Jeffery (1999) used human judgment to determine the feature weights, Angelis and Stamelos (2000) determined the feature weight by some statistics (such as inverse variance or range values), Mendes *et al.* (2003) employed a statistically significant correlation approach for the assignment of feature weights, Auer *et al.* (2006) developed a flexible exhaustive search method to determine the weights, Most recently, Huang and Chiu (2006) proposed the genetic algorithm for feature weighting.

Historical Database

The historical database used for retrieving the similar past projects is also a key component in ABE system. Reducing the whole historical data set into a smaller subset that consists only of representative projects can significantly improve ABE's performance. First, it reduces the search space and therefore can save computational time searching for nearest neighbors. Second, it produces quality results because it may eliminate some outliers in the dataset. However, very list research has been done on this topic. Kadoda *et al.* (2000) conducted one preliminary study on project selection via forward sequential selection. Recently, Li *et al.* (2007, 2009b) proposed a genetic algorithm based framework to optimize both project subset and feature weights.

Number of Nearest Neighbors: K

The number of nearest neighbors *K* is another important parameter in ABE system. Many papers have investigated the impacts of this parameter on estimation results or considered optimizing this number (See table 2.3).

Table 2.3: Summary of papers investigating different number of nearest neighbors

Source	# of nearest neighbors (*K*)	Method of selecting *K*	Comments
Shepperd and Schofield (1997)	1-3	Cross validation	No one approach is consistently more accurate so the decision requires a certain amount of experimentation on the part of the estimators.
Walkerden and Jeffery (1999)	1	Predefine	N.A.
Angelis and Stamelos (2000)	1-23	Cross validation	No rule to decide this number without experiments
Leung, 2002	2	Predefine	Two nearest neighbors have a higher referencing value and overcome some problems with one nearest neighbor.
Mendes et al., 2003	2,3	Predefine	N.A.
Jorgensen et al. (2003)	1-3	Predefine	The best performance was achieved using the closest analogue. Inclusion of two or three analogues did not improve the accuracy.
Auer et al. (2006)	1	Predefine	N.A.
Huang and Chiu (2006)	1-3	Cross validation	The use of closest two or three analogues presents better result than use closest analogue.
Chiu and Huang (2007)	1	Predefine	The decision is case-to-case since no heuristic method currently exists.
Li et al. (2007)	1-80	Cross validation	Jack-knife for optimizing K and T.
Mittas et al. (2008)	1-10	Cross validation	N.A.
Li and Ruhe (2008a)	1-80	Dynamic *K*	N.A.
Li and Ruhe (2008b)	1-80	Dynamic *K*	N.A.

Five of the previous studies have specified a certain range for *K* values, and then applied the cross-validation procedure to select the optimal *K* value with which the ABE could produce the predictions optimizing the error metric on the training dataset. Moreover, six papers (Walkerden and Jeffery 1999, Leung 2002, Mendes et al., 2003, Jorgensen et al. 2003, Auer *et al.* 2006, Chiu and Huang 2007) predefine *K* at fixed values. More recently, Li and Ruhe (2008a, 2008b) adopted a method named 'dynamic *K*' which was first proposed by Kadoda *et al.* (2000). In this approach, the projects falling within a certain distance threshold (*T*) of the target project are treated as nearest neighbors and the number of neighbors may vary when different target projects appear. This method can also be regarded as one cross validation scheme. The advantage of cross validation is that it takes into account the information from dataset.

The Solution Function

The final prediction for the new project is produced by the solution function based on the selected *K* nearest neighbors. The solution function has the following general form.

$$\hat{C}_x = g(C_1, C_2, \ldots, C_K) \tag{2.12}$$

where $\hat{C}_x$ is the estimation value, C_i is the cost value of *i*th nearest neighbor, and K denotes the number of nearest neighbors. Different types of solution functions have been proposed for more accurate estimating results. Table 2.4 summarizes the published solution functions. The solution functions presented in table 2.4 are: the un-weighted mean, the median, and the inverse distance weighted mean.

Table 2.4: Summary of publications with different solution functions

Source	Un-weighted mean	Weighted mean	Median	Adjustment
Shepperd and Schofield (1997)	Yes	Yes	No	No
Walkerden and Jeffery (1999)	Yes*	No	No	Yes
Angelis and Stamelos (2000)	Yes	No	Yes	No
Leung (2002)	Yes	No	No	Yes
Mendes et al. (2003)	Yes	Yes	Yes	No
Jorgensen et al. (2003)	Yes*	No	No	Yes
Auer et al. (2006)	Yes*	No	No	No
Huang and Chiu (2006)	No	Yes	No	No
Chiu and Huang (2007)	Yes*	No	No	Yes
Li et al. (2007)	No	Yes	No	No
Mittas et al. (2008)	No	Yes	No	Yes
Li and Ruhe (2008a)	No	Yes	No	No
Li and Ruhe (2008b)	No	Yes	No	No
Totals	4+4*	7	2	5

* means only one nearest project is used

The un-weighted mean is the simple average of the cost values of K nearest neighbors, where $K > 1$. It is a classical measure of central tendency and treats all most similar projects as being equally influential on the cost estimates.

The median is the median of the cost values of K nearest neighbors, where $K > 2$. It is another measure of central tendency and it is a more robust statistic when the number of nearest neighbors increases (Angelis and Stamelos, 2000).

The inverse distance weighted mean (Kadoda, *et al* 2000) allows more similar projects to have more influence than less similar ones. The formula for weighted mean is shown in (2.13):

$$\hat{C}_p = \sum_{k=1}^{K} \frac{Sim(p, p_k)}{\sum_{i=1}^{n} Sim(p, p_k)} C_{p_k} \tag{2.13}$$

where p denotes the new project being estimated, p_k represents the kth nearest neighbors, $Sim(p, p_k)$ is the similarity between project p_k and p, C_{p_k} is the cost value of p_k, and K is the total number of nearest neighbors.

The weighted mean has become more and more popular in recent years (except those using unique nearest project). This might be due to the fact that the weighted mean allows the more similar projects to have more influence than the lower ones (Huang and Chiu 2006). However, in light of our knowledge there is no solid evidence or proof that supports this argument.

In addition, the last column of table 2.4 presents the studies on the adjustments to the solution functions. The adjustment on the solution function is necessary since it can capture the differences between the new project and the retrieved projects, and refine the retrieved solution into the target solution (Walkerden and Jeffery, 1999). Many researchers have proposed different techniques to adjust the solution function. Leung (2002) proposed a refinement which is based on the relative location of the target project. Jorgensen *et al.* (2003) proposed Regression Toward the Mean (RMT) adjusting the analogy based results:

$$\begin{aligned} \hat{C}_x &= FP \times \hat{P}_x \\ \hat{P}_x &= P_k + (M - P_k) \times (1 - r) \end{aligned} \tag{2.14}$$

where C_x denotes the cost of a new project x, $\hat{P}_x$ denotes the adjusted productivity (productivity = cost/FP) of the new project, P_k is the average of the *k* nearest neighbors, *M* is the mean productivity of the similar projects, and *r* is the correlation coefficient between the productivity of closest analogues and the actual productivity.

More recently, Chiu and Huang (2007) proposed an additive adjustment.

$$\hat{C}_x = C_1 + Adj \tag{2.15}$$

where C_1 is the cost value of the first nearest neighbor to x and $Adj = \sum_{i=1}^{n} \alpha_i \times (f_{xi} - f_{1i})$ is the adjustment term. f_{xi} is the *i*th feature of the new project x. f_{1i} denotes the *i*th feature of the nearest project. Then GA is applied to optimize the coefficients α_i. In a more recent work, Mittas et al. (2008) introduced the iterated bagging (Bootstrap aggregating) technique to adjust the solution function. The bagging predictor is defined as:

$$\hat{C}_x = \frac{1}{T}\sum_{t=1}^{T} C_1^{(t)} \tag{2.16}$$

where $C_1^{(t)}$ is the nearest neighbor cost value obtained on the bootstrap sample $t = 1,\ldots, T$. T is the total number of bootstrapped samples.

We can tell from the above that most previous works were focusing on linear type of adjustments. Since non-linearity is a common characteristic throughout the software engineering data sets, the non-linear type of adjustment is of great practical importance for investigation.

2.4 Evaluation Criteria

Evaluation criteria are essential for the empirical validations of cost estimation methods. To measure the accuracy of estimation methods, various kinds of evaluation criteria have been developed. In this section, we collect 19 different criteria appeared in the publications of the past decade. Among all evaluation criteria, the Mean Magnitude of Relative Error (MMRE) and Prediction at level q (PRED (q)) (Conte *et al.* 1986) are most frequently used. Although MMRE is biased and not always reliable as a performance metric, it still has been the de facto standard in the software cost estimation literature.

In addition, other types of evaluation criteria have also been used. For instance, Kemerer (1987) introduced R-Square as an error metric for cost estimation. Miyazaki *et al.* (1994) proposed the use of the Balanced Mean Magnitude of Relative Error (BMMRE) as well as the Inverted Balanced Mean Magnitude of Relative Error (IBMMRE). Jorgensen *et al.* (1995) introduced the Median Magnitude of Relative Error (MdMRE). Lo and Gao (1997) proposed two error metrics: the Weighted Mean of Quartiles of relative errors (WMQ) and the Standard Deviation of the Ratios of the estimate to actual observation (SDR). The Mean Magnitude of Error Relative to the estimate (MMER) or Mean Variation From Estimate (MVFE) as a different name, are introduced by Kitchenham *et al.* (2001), and Hughes *et al.* (1998) respectively. Kitchenham *et al.* (2001) also suggested Mean of the Absolute Residual (MAR) and Median of the Absolute Residual (MdAR) as the candidate metrics.

The development of new metrics is an ongoing process. More recent metrics that have been introduced or proposed are: Adjusted Mean Square Error (AMSE) (Burgess and Lefley 2001), Standard Deviation (SD) (Foss *et al.* 2003), Relative Standard Deviation (RSD) (Foss *et al.* 2003), Logarithmic Standard Deviation (LSD) (Foss *et al.* 2003), and Logarithmic Relative Error (LRE) (Jorgensen 2004).

Based on the mathematical structure of the error metrics, we classify all of them into four different groups: relative error based metrics, absolute error based metrics, sum of squares errors based metrics, and ratio error based metrics.

2.4.1 Relative Error based Metrics

The error metrics in this group are based on the relative error named Magnitude of Relative Error (MRE$_i$) (Conte *et al.* 1986):

$$MRE_i = \left| \frac{C_i - \hat{C}_i}{C_i} \right| \tag{2.17}$$

where C_i denotes the actual cost of the *i*th project, and $\hat{C}_i$ denotes the estimated effort of *i*th project.

Mean Magnitude of Relative Error (MMRE) (Conte et al. 1986)

The MMRE is defined as:

$$MMRE = \frac{1}{n} \times \sum_{i=1}^{n} \left| \frac{C_i - \hat{C}_i}{C_i} \right| = mean(MRE) \tag{2.18}$$

where *n* denotes the number of projects being estimated, C_i denotes the actual effort of *i*th project, and $\hat{C}_i$ denotes the estimated effort of *i*th project. Small MMRE value indicates a low level of estimation error. However, this metric is unbalanced and it penalizes overestimation more than underestimation.

PREDiction at level q (PRED (q)) (Conte *et al.* 1986)

The PRED is the percentage of predictions that fall within a specified percent of the actual cost:

$$PRED(q) = \frac{1}{n} \times \sum_{i=1}^{n} \delta(MRE_i - q)$$
$$\delta(x) = \begin{cases} 1, & x \le 0 \\ 0, & x > 0 \end{cases} \quad (2.19)$$

where q is a predefined threshold. PRED(q) computes the percentage of the predictions whose MRE values are less than or equal to q. In most publications, q is set to 0.25 or 0.30.

Mean of weighted qualities (MWQ) (Lo and Gao 1997)

MWQ is the weighted mean of MREs:

$$MWQ = \frac{Q_1 + 2Q_2 + 3Q_3}{6} \quad (2.20)$$

where Q_1 is the first quartile, Q_2 is the second quartile and Q_3 is the third quartile of the MREs. The smaller the MWQ is, the more accurate the estimation is. It is shown that MWQ and MMRE are consistent when there are no outliers and the estimation is unbiased (Lo and Gao 1997).

Median Magnitude of Relative Error (MdMRE) (Jorgensen *et al.* 1995)

MdMRE is the median of the *MRE*s:

$$MdMRE = median(MRE) \tag{2.21}$$

It is an aggregate measure and compared to MMRE it is less sensitive to extreme values (Foss *et al.* 2003).

Balanced Mean Magnitude of Relative Error (BMMRE) (Miyazaki *et al.* 1994)

To overcome the drawback of MMRE that penalizes overestimation more than underestimation, Miyazaki *et al.* (1994) proposed the so called balanced MMRE.

$$BMMRE = \frac{1}{n} \times \sum_{i=1}^{n} \left| \frac{C_i - \hat{C}_i}{\min(C_i, \hat{C}_i)} \right| \tag{2.22}$$

Inverted Balanced Mean Magnitude of Relative Error (IBMMRE) (Miyazaki *et al.* 1994)

This metric is similar to BMMRE, the only difference is that the denominator of IBMMRE is the maximum of real cost and predicted cost:

$$IBMMRE = \frac{1}{n} \times \sum_{i=1}^{n} \left| \frac{C_i - \hat{C}_i}{\max(C_i, \hat{C}_i)} \right| \tag{2.23}$$

Mean Magnitude of Error Relative to the estimate (MMER) (Kitchenham *et al.* 2001) or *Mean Variation From Estimate (MVFE)* (Hughes *et al.* 1998)

$$MMER = \frac{1}{n} \times \sum_{i=1}^{n} \left| \frac{C_i - \hat{C}_i}{\hat{C}_i} \right| \tag{2.24}$$

This metric is proposed because project managers are normally aware of the estimated cost for a project before the actual cost and a measurement based on the ratio |actual-estimate|/estimate would seem to be a more accurate reflection of managerial concerns (Hughes *et al.* 1998).

2.4.2 Absolute Error based Metrics

In the literature, there are only two metrics based on the absolute error. These metrics seem to be less popular than the relative error based ones. The reason might be that the cost values of software projects often vary a lot and it is difficult to use the absolute errors to compare one group of projects with very small costs against another group of projects with large costs. However, as the absolute error is a balanced metric, many studies (Mendes et al. 2003, Li et al. 2009) also use it to diagnose the conclusion made by relative errors.

Mean of the Absolute Residual (MAR) (Kitchenham *et al.* 2001)

MAR is the mean value of the absolute errors:

$$MAR = \frac{1}{n} \times \sum_{i=1}^{n} \left| C_i - \hat{C}_i \right| \tag{2.25}$$

Median of the Absolute Residual (MdAR) (Kitchenham *et al.* 2001)

MdAR is the median value of the absolute errors:

$$MdAR = median(\left| C_i - \hat{C}_i \right|) \tag{2.26}$$

2.4.2 Sum of Square Errors based Metrics

Sum of square errors (or mean of square errors) is often used by statisticians to measure the errors. Many studies using statistical techniques such as regressions consider the sum of square errors based metrics, especially R-square.

Root Mean Square (RMS) (Conte *et al.* 1986)

It is the root of the mean square error:

$$RMS = \sqrt{\frac{\sum_{i=1}^{n} (C_i - \hat{C}_i)^2}{n}} \tag{2.27}$$

Relative Root Mean Square (RRMS) (Conte *et al.* 1986)

It is the RMS divided by the mean of the actual costs:

$$RRMS = RMS \Big/ \frac{\sum_{i=1}^{n} C_i}{n} \tag{2.28}$$

Adjusted Mean Square Error (AMSE) (Burgess and Lefley 2001)

It is the sum of the squared errors divided by the product of the actual and the estimated cost:

$$AMSE = \sum_{i=1}^{n} \frac{(C_i - \hat{C}_i)^2}{C_i \times \hat{C}_i} \tag{2.29}$$

R-Square & Adjusted R-square **(**Kemerer 1987)

R-square and Adjusted *R*-square indicate the percentage of total variation explained by the regression model. They are the common measures of the regression's goodness of fit. *R*-square has the following mathematical expression:

$$R^2 = \frac{\sum_{i=1}^{n} (\hat{C}_i - \overline{C}_i)^2}{\sum_{i=1}^{n} (C_i - \overline{C}_i)^2} \tag{2.30}$$

where $\overline{C_i}$ is the mean of the real cost values C_i. *R*–square describes the percentage of total variance explained by the model. A high *R*-square value indicates a good model fit with observed data. However, the *R*-square also increases along with the number of explanatory variables in the linear equation even though these variables are not significant in explaining the variability of the dependent variable. Therefore, the adjusted *R*-square is used.

$$R_{adj}^2 = 1 - \frac{n-1}{n-p} \times (1 - R^2) \tag{2.31}$$

where n is the number of projects in the datasets and p is the number of explanatory variables.

Standard Deviation (SD) (Foss *et al.* 2003)

The standard deviation is a common metric to evaluate the variance of the predicted values:

$$SD = \sqrt{\frac{\sum_{i=1}^{n}(C_i - \hat{C}_i)^2}{n-1}} \tag{2.32}$$

Relative Standard Deviation (RSD) (Foss *et al.* 2003)

RSD is modified from SD and it incorporates the size of the project:

$$RSD = \sqrt{\frac{\sum_{i=1}^{n} \left(\frac{C_i - \hat{C}_i}{x_i}\right)^2}{n-1}} \tag{2.33}$$

where the variable x is the number of function points (FP). The rationale behind RSD is to measure the dispersion relative to the x value (e.g., FP) rather than relative to the C value to avoid one of the problems with MMRE. One of MMRE's problems is that small actual costs (small Cs) will have a (too) large influence on the mean MRE since a number divided by a small number tends to be a large number. Contrary to MRE, which is almost uncorrelated with size, SD is positively correlated with size because software project data sets are often heteroscedastic. As opposed to SD, RSD is almost uncorrelated with size. We observe that RSD is limited to models with a single predictor variable. In many software studies, this is, however, not a serious limitation since it is common to create prediction models based on FP and effort. More importantly, we can provide a rationale for choosing this metric as well as an interpretation of its meaning.

Logarithmic Standard Deviation (LSD) (Foss *et al.* 2003)

$$LSD = \sqrt{\frac{\sum_{i=1}^{n} \left(e_i + \frac{s^2}{2}\right)^2}{n-1}}$$

$$e_i = \ln C_i - \ln \hat{C}_i \tag{2.34}$$

The term s^2 is an estimator of the variance of the residual e_i. The rationale behind LSD is as follows: Data sets with a large heteroscedasticity will be very much influenced by the large projects. Thus, the usual SD is more sensitive to large projects than to small projects and it may therefore not be a stable, reliable measure for such data sets. On the other hand, LSD lends itself well to data sets that comply with a log-linear model because the residual error is independent of size (i.e., homoscedastic) on the log scale.

2.4.4 Ratio Error based Metrics

Because MRE is an unbalanced metric, many authors consider the alternatives based on the ratio error metrics. The ratio error is defined as follows:

$$R_i = \frac{\hat{C}_i}{C_i} \tag{2.35}$$

where C_i is the actual cost/effort and $\hat{C}_i$ is the estimated cost/effort.

Logarithmic Relative Error (LRE) (Jorgensen 2004c)

LRE is the absolute value of the logarithm of the ratio error:

$$LRE = \left|\ln(R_i)\right| \tag{2.36}$$

Standard Deviation of R (SDR) (Lo and Gao 1997)

SDR measures the estimation consistency of R_{i}:

$$SDR = \sqrt{\frac{\sum_{i=1}^{n}(R_i - \overline{R})^2}{n-1}} \tag{2.37}$$

where $\overline{R}$ is the mean of the ratio R_{i}. The smaller the SDR, the more consistent the estimation is (Lo and Gao 1997).

Chapter 3 Feature Selection Based on Mutual Information[1]

As mentioned in the previous chapter, feature selection is an important preprocessing stage of analogy based estimation. Most existing feature selection methods of ABE are 'wrappers' which can usually yield high fitting accuracy at the cost of high computational complexities and poor explanations of the selected features. In this chapter, the mutual information based feature selection technique (MIABE) is proposed. This approach hybridizes both 'wrapper' and 'filter' mechanism. 'Filters' are another type of feature selectors with much lower computation complexity and more interpretable resulting features, though they may not produce the fitting results as accurate as 'wrappers' do. The MIABE is compared with several established feature selectors. The results show that MIABE is an effective feature selector which can produce quality predictions with low computational cost and explainable features.

[1] This chapter is relevant to the publication Li et al. 2009a

3.1 Introduction

The fundamental principle of ABE is simple: given a new project for estimation, the most similar historical projects are selected to predict the cost of the new project by using a similarity measure. As one of the key components of ABE, the similarity measure is used to aggregate the similarity under each project feature (or cost driver). As shown in section 2.3.5, the choice of project features has large impact on the estimation results. The feature selection is proposed to determine the optimum subset of features that give the most accurate estimation (Mendes *et al.*, 2003).

In software cost estimation literature, some feature selection methods have been proposed, such as exhaustive search (Shepperd and Schofield, 1997), hill climbing and forward sequential selection (Kirsopp *et al.*, 2002). However, most existing feature selectors are the so called 'wrappers' (Kohavi and John, 1997). The 'wrappers' convolve with ABE method, with the direct goal to minimize the fitting error of the particular problem. Usually, 'wrappers' can yield high fitting accuracy at the cost of high computational complexity and low interpretations of the selected features. It is fairly possible that less informative features lead to poorer prediction accuracy.

To address these issues, in this chapter we propose a novel feature selection algorithm combining wrapper mechanism and filter mechanism (Almuallim and Dietterich, 1994, Kohavi and John, 1997). Unlike 'wrappers', the filter mechanism selects features by evaluating some preset criteria independently of the fitting accuracy of ABE. In general, the filter approach has much lower complexity than wrappers, and the features selected by 'filters' are more interpretable, which in turn could generate more accurate predictions (Peng *et al.*, 2005).

In filter mechanism, we choose mutual information (MI) (Battiti, 1994, Kwak and Choi, 2002(a), Kwak and Choi, 2002(b), Peng *et al.*, 2005) as the preset criterion. The reasons to consider mutual information are: 1) it is capable of measuring arbitrary relations (include both linear and non-linear) between features, 2) it is independent of the transformations (such as normalization and scaling) acted on features (Battiti, 1994). Based on mutual information criterion, we propose mutual information based feature selection approach for analogy based estimation (MIABE). MIABE adopts filter mechanism in the inner stage and the wrapper mechanism in the outer stage. The inner stage selects the feature subsets maximizing mutual information between the selected features and the target feature (software cost). The outer stage searches for the feature subset maximizing the fitting accuracy from the candidate feature subsets generated by the inner stage.

The rest of this chapter is organized as follows: section 3.2 presents the concepts related to mutual information, the algorithms to calculate mutual information and the proposed MIABE algorithm. Section 3.3 presents the experiment setup of this study. Section 3.4 describes two numerical examples and the analysis of the results.

3.2 Mutual Information Based Feature Selection for Analogy Based Estimation

A number of selection criteria, such as correlation coefficient and least square regression error, are available for the filter mechanism for feature selection. In our study, mutual information (MI) (Shannon and Weaver, 1949) is chosen as the selection criterion because MI is capable of measuring a general dependence between two features without assuming the distributions of the features. This capability of MI matches one important property of ABE: ABE requires no assumption on the distributions of features to derive the solutions (Walkerden and Jeffery, 1999). In addition, MI is capable to manage both numerical and categorical features which often simultaneously appear in software engineering datasets.

In section 3.2.1, we briefly introduce basic concepts and notations of the theory related to MI. In section 3.2.2, the calculation of MI is discussed. In section 3.2.3, the MIABE approach is presented.

3.2.1 Entropy and Mutual Information

In feature selection problem, the relevant features have important information regarding the output of ABE, whereas the irrelevant features contain little information regarding the output of ABE. The objective of feature selection is to find those features that contain as much information about the output as possible. For this purpose, Shannon’s information theory

(Shannon and Weaver, 1949) provides a feasible way to measure the information by entropy and mutual information.

The entropy $H(X)$ is a measure of the uncertainty of a random variable X. For a discrete random variable X, with the probability mass function $p(x)$, the entropy of X is defined as:

$$H(X) = -\sum_{x \in \Phi} p(x) \log p(x) \tag{3.1}$$

where Φ is the sample space of variable X, and the logarithm is based on 2. Information entropy is expressed in bits. The joint entropy of X and Y with joint pdf: $p(x, y)$ is defined as follows:

$$H(X,Y) = -\sum_{x \in \Phi} \sum_{y \in \Omega} p(x, y) \log p(x, y) \tag{3.2}$$

where Ω is the sample space of variable Y, When certain variables are known and other variables are unknown, the remaining uncertainty is measured by the conditional entropy:

$$\begin{aligned} H(Y \mid X) &= -\sum_{x \in \Phi} p(x) H(Y \mid X = x) \\ &= -\sum_{x \in \Phi} \sum_{y \in \Omega} p(x, y) \log p(y \mid x) \end{aligned} \tag{3.3}$$

From formulae (3.1), (3.2) and (3.3), the joint entropy and conditional entropy have the following relation:

$$H(X,Y) = H(X) + H(Y \mid X) = H(Y) + H(X \mid Y) \tag{3.4}$$

Based on definitions about entropy, the mutual information (MI) between two variables is defined as below:

$$I(X;Y) = \sum_{x \in \Phi} \sum_{y \in \Omega} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \tag{3.5}$$

If the mutual information is large, the two variables *X* and *Y* are closely related, while if the mutual information becomes zero, the two variables *X* and *Y* become independent. The mutual information and the entropy have the following relationships:

$$\begin{aligned} &I(X;Y) = H(X) - H(X \mid Y) \\ &I(X;Y) = H(Y) - H(Y \mid X) \\ &I(X;Y) = H(X) + H(Y) - H(X,Y) \\ &I(X;Y) = I(Y;X) \\ &I(X;X) = H(X) \end{aligned} \tag{3.6}$$

An illustrative presentation of the relationships is given in fig 3.1. The mutual information corresponds to the intersection part between the entropy of *X* and the entropy of *Y*.

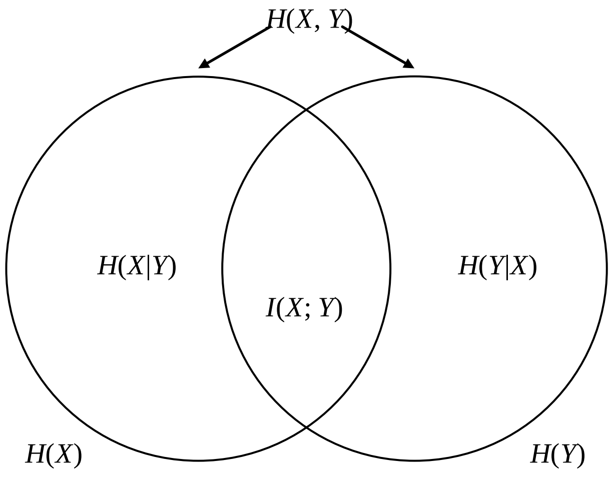

Figure 3.1: The relations between mutual information and the entropy

So far the concepts of entropy and mutual information are introduced for the discrete variables. But in software engineering databases many software project features are continuous in nature. For continuous variables the entropy and mutual information are defined as follows:

$$\begin{aligned} H(X) &= -\int p(x)\log p(x)dx \\ I(X;Y) &= \iint p(x,y)\log\frac{p(x,y)}{p(x)p(y)}dxdy \end{aligned} \qquad (3.7)$$

$H(X)$ and $I(X;\,Y)$ of continuous variables have the same properties as what the $H(X)$ and $I(X;\,Y)$ of discrete variables have in (3.6). However, when the underlying probability density functions ($p(x)$, $p(y)$, and $p(x,\,y)$) are continuous, it is often impossible to obtain the analytical integration. Therefore, approximation methods have been proposed (Moddemeijer, 1989, Kwak and Choi 2002(b)).

3.2.2 Mutual Information Calculation

Continuous software project features such as project cost, lines of code, function points, often appear in software engineering datasets. However, the approximation of MI between continuous variables is difficult. One possible solution is the traditional histogram approach (Moddemeijer, 1989) which involves discretizing the data into equally sized intervals. Although the histogram approache can obtain satisfactory results under low-dimensional conditions, the accuracy of most histogram estimations is substantially degraded when high dimensional data appears (Fraser and Swinney, 1986). An alternative solution is using the continuous kernel based density estimator to approximate $I(X; Y)$, as proposed by (Kwak and Choi 2002(b)).

In this method, given N samples of a random variable X, the approximate density function has the following form:

$$\hat{p}(x) = \frac{1}{N}\sum_{i=1}^{N}\delta(x - x^{(i)}, h) \tag{3.8}$$

where $\delta(\bullet)$ is the Parzen window function (Parzen, 1962), $x^{(i)}$ is the ith sample, and h is the window width. Parzen has proven that with the properly chosen $\delta(\bullet)$ and h, the estimation $\hat{p}(x)$ can converge to the true density $p(x)$ when N tends to infinity. Usually, $\delta(\bullet)$ is chosen as the Gaussian window:

$$\delta(z,h) = \frac{1}{(2\pi)^{d/2} h^d \left|\sum\right|^{1/2}} \exp(-\frac{z\sum^{-1} z}{2h^2}) \tag{3.9}$$

where $z = x - x^{(i)}$, d is the dimension of the sample x and $\sum$ is the covariance of z. In our study, the calculation is accomplished by using Peng's 'Mutual Information computation' MATLAB package available online (Peng 2007).

3.2.3 Mutual Information Based Feature Selection for Analogy Based Estimation

In this section, the proposed algorithm for feature selection using mutual information (MIABE) is presented. The algorithm consists of two stages: the inner stage, at which the classical MIFS algorithm (Battiti, 1994) is used to select out m number of features, and the outer stage, which determines the value of m by minimizing the fitting error of ABE on training dataset. The system diagram of the proposed feature selection approach is shown in fig. 3.2.

In the inner stage, the classical MIFS algorithm is performed by the following procedure:

1) (Initialization) Set F = 'whole feature set', S = 'empty set', let C represents the cost value, f_i represents the ith project feature.

2) $\forall f_i \in F$, compute $I(C; f_i)$.

3) Find the feature f_i that maximizes $I(C; f_i)$, set $F \leftarrow F \backslash \{ f_i \}$, $S \leftarrow \{f_i\}$.

4) (Greedy selection) repeat until desired number m of features are

selected.

a) (Computation of the MI between features) for all couples of features $(f_i,\ f_s)$, $f_i \in F$, $f_s \in S$, compute $I(f_i; f_s)$, if it is not yet available.

b) (Selection of the next feature) choose the feature $f_i \in F$ that maximizes $I(C; f_i) - \beta \sum_{f_s \in S} I(f_i; f_s)$; set $F \leftarrow F \backslash \{ f_i \}$, $S \leftarrow \{f_i\} \cup S$.

5) Output the subset S containing m selected features. m will be optimized in the outer stage by maximizing ABE's fitting accuracy.

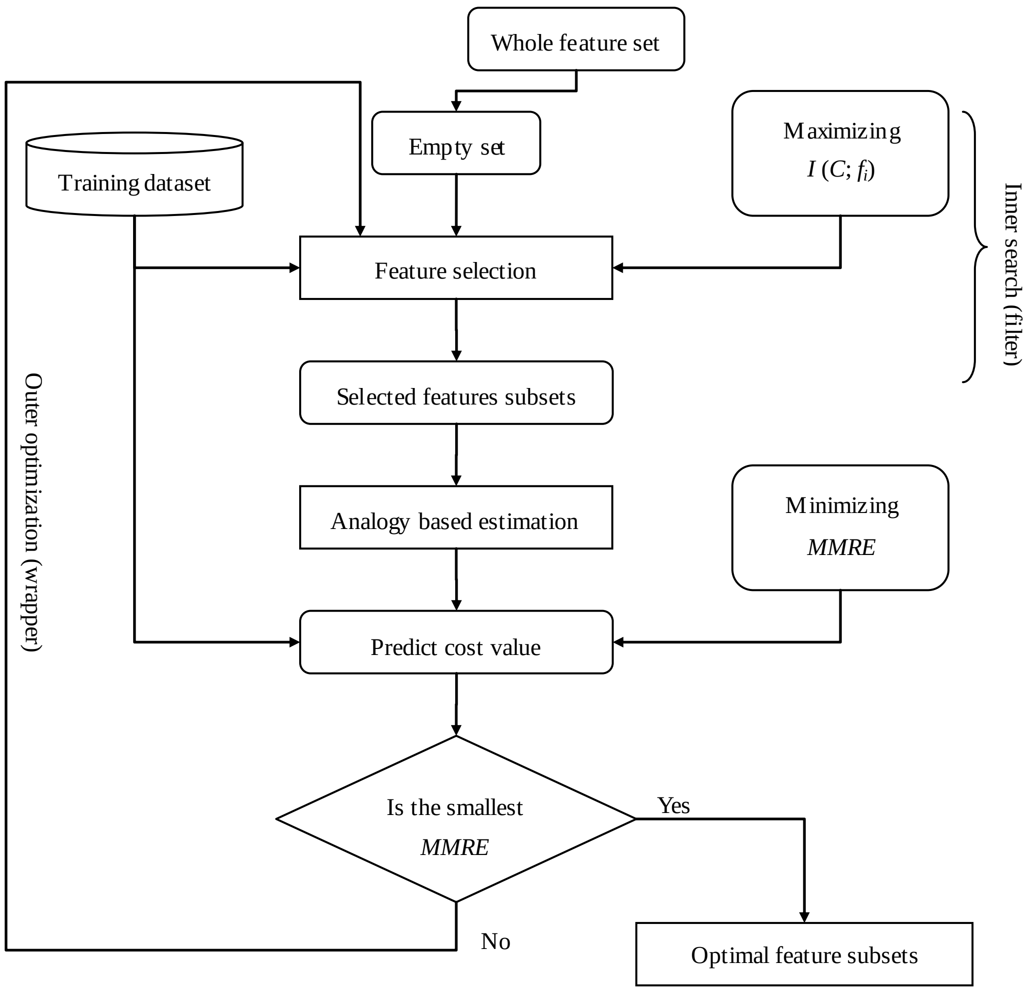

Figure 3.2: The schematic diagram of proposed MIABE algorithm

The parameter β controls the redundancy among the features. If β is zero, the mutual information among features is not taken into consideration and the algorithm selects features in the order of the mutual information between features and project cost. As β grows, the mutual information among features begins to influence the selection process and the redundancy among features is reduced. However, if β is too large, the feature-cost relation will be overwhelmed by the relations among the features. In this study β is set to 0, because only the feature-cost relation is of interest and the computation of β demands extra computational resources.

The outer stage solves the remaining issue in the inner stage: determining the optimal number of features m. Suppose that there are a total number of n features in the dataset. The MIFS is used to select 1 to n features and this process leads to n sequential feature sets $S_1 \subset S_2 \subset ... \subset S_m \subset ... \subset S_{n-1} \subset S_n$. Then compare all the n sequential feature sets $S_1,\ldots,S_m,\ldots, S_n$ to find the set S_m with the minimal training error of ABE (MMRE is used to measure training error in this study). Therefore, m is the optimal number of features and S_m is the optimal feature set.

3.3 Experiment Design

The evaluation criteria and data sets for experiments are presented in section 3.3.1 and section 3.3.2 respectively.

3.3.1 Evaluation Criteria

Evaluation criteria are essential to the experiments. In the literature, several quality metrics have been proposed to assess the performances of estimation methods. More specifically, Mean Magnitude of Relative Error (MMRE), PRED(0.25) (Conte *et al.* 1986), and Median Magnitude of Relative Error (MdMRE) (Jorgensen *et al.* 1995) are three popular metrics.

The MMRE is as defined below:

$$\begin{aligned} MMRE &= \frac{1}{n} \times \sum_{i=1}^{n} MRE_i \\ MRE_i &= \left| \frac{C_i - \hat{C}_i}{C_i} \right| \end{aligned} \tag{3.10}$$

where *n* denotes the totoal number of projects, C_i denotes the actual cost of project *i*, and $\hat{C}_i$ denotes the estimated cost of project *i*. Small MMRE value indicates a low level of estimation error. However, this metric is unbalanced and penalizes overestimation more than underestimation.

The MdMRE (Kitchenham *et al.* 2001) is the median of all the MREs.

$$MdMRE = median\left(\left| \frac{C_i - \hat{C}_i}{C_i} \right|\right) \tag{3.11}$$

It exhibits a similar pattern to MMRE but it is more likely to select the true model especially in the underestimation cases since it is less sensitive to extreme outliers (Foss *et al.*, 2003). The PRED(0.25) is the percentage of predictions that fall within 25 percent of the actual value.

$$PRED(0.25) = \frac{1}{n} \times \sum_{i=1}^{n} \delta(MRE_i - 0.25)$$
$$\delta(x) = \begin{cases} 1, & x \le 0 \\ 0, & x > 0 \end{cases} \tag{3.12}$$

The PRED(0.25) identifies cost estimations that are generally accurate, while MMRE is a biased and not always reliable as a performance metric. However, MMRE has been the de facto standard in the software cost estimation literature. In addition to the metrics mentioned above, there are several metrics available in the literature. Interested readers can refer to section 2.4 for more information.

3.3.2 Data Sets

Two representative datasets are selected for experiments. They are Desharnais dataset (Desharnais 1989) containing merely numerical features and Maxwell dataset (Maxwell, 2002) mainly composed of categorical features.

Despite the fact that Desharnais dataset is quite old, it is still one of the large and publicly available datasets. Therefore it still has been used by many recent research works, such as Mair *et al.* (2000), Burgess and Lefley, (2001), and Auer *et al.* (2006). This data set includes 9 numerical features and 81 projects. Four out of 81 projects have been excluded due to the missing feature values. This process results in the 77 complete projects for experiments (Mair *et al.* 2000). The eight independent features of this data set, namely "TeamExp", "ManagerExp", "YearEnd", "Transactions", "Entities", "PointsAdjust", "Envergure", and "PointsNonAjust" are all considered for constructing the models. The dependent feature "Effort" is measured in 1000 *h*. The detailed definitions and descriptive statistics of all the features are shown in table B.3 and table B.4 of Appendix B.

The Maxwell dataset with 62 projects' data from one of the biggest commercial banks in Finland is a relatively new software project datasets and has been used by some recent research works (Maxwell, 2002, Sentas *et al.*, 2005). The detailed definitions and descriptive statistics of all the features are shown in table B.5 and table B.6 of Appendix B. Most features in this dataset are categorical and the numerical features are time, duration, size and effort. The categorical features can be further classified into ordinal features and nominal features. The ordinal feature and nominal feature have to be distinguished while calculating the similarity measure (See formula (2.10) and (2.11)).

The ordinal features are 'nlan', and 't01'-'t15', while the nominal features are 'app', 'har', 'dba', 'ifc', 'source' and 'telonuse'. By following Maxwell's process, in our analysis we used the new features 'subapp' and 'subhar', instead of the features 'app' and 'har'. These new features are subsets of the original ones and they contain categories with 3 or more observations. More specifically, the levels of 'subapp' are 1, 2, 3 and 5 and the levels of 'subhar' are 1, 2, and 5.

3.3.3 Experiment Design

Before the experiments, all types of features (numerical, ordinal, and nominal) in the two data sets are normalized into [0, 1] in order to eliminate the different influences of the features. For the purpose of comparing our method to the published works, different validation schemes are applied.

For Desharnais dataset, our method is compared to Mair's work (Mair *et al.* 2000) where methods are trained and tested by the three-folder cross-validation. This cross-validation yields three different training-testing set combinations. Each testing set is randomly generated from the original dataset and the remaining projects are used as the training set. Therefore, for the Desharnais datasets we obtain three different training splits and three testing splits. By following the splitting scheme of Mair *et al.* (2000), the 87% split (87% in the training set and 13% in the testing sets) is used. The training set is only used to develop the estimating methods, while the testing set is exclusively used to test the estimation performance of the candidate methods. The accuracies across all training splits are aggregated as the training results, and the accuracies across all testing splits are aggregated as the testing results.

For Maxwell dataset, our method is compared to Maxwell's work in

(Maxwell, 2002) and Sentas *et al.*'s work in (Sentas *et al.* 2005). Therefore, we prepare the training and testing datasets by following their splitting method. The 50 projects that completed before 1992 form the training set, and the 12 projects that finished from 1992 to 1993 are used as testing set.

After determining the validation scheme, the following experiment procedures are conducted on two datasets.

- Firstly, the performances of MIABE feature selection are investigated by fixing the ABE parameter settings. As mentioned in last chapter, ABE has three controllable parameters: similarity measure, number of nearest neighbors *K*, and solution function (historical dataset is excluded from consideration). In line with Kirsopp *et al.* (2002)'s setting, these parameters are fixed to Euclidean distance, $K = 3$ and inverse distance weighted mean respectively. Then MIABE is compared against other wrapper feature selection methods: Exhaustive selection (EX) (Shepperd and Schofield 1997), Hill Climbing feature selection (HC), and Forward Sequential feature selection (FSS) (Kirsopp *et al.* 2002). The implementations of these methods are realized by using the automatic tool archANGEL. The configurations of HC and FSS follow Kirsopp *et al.* (2002).
- Secondly, the features that selected by feature selection methods are analyzed by mutual information diagrams. The MI diagram is a useful tool for diagnosing the feature selection phase.

- Thirdly, the computational efficiencies of the feature selection methods are tested and compared.
- Finally, the MIABE's three parameters are optimized and the optimal MIABE is compared with the published works.

The results and analyses are presented in the following two sections.

3.4 Results

To validate the proposed MIABE method, this section summarizes the results of two empirical studies on the two datasets described in Section 3.3.2.

3.4.1 Results on Desharnais Dataset

The experimental results on Desharnais data are presented in following paragraphs. Fixing the three parameters (similarity measure, number of nearest neighbors *K*, and solution function) of ABE method, we first compare MIABE with three 'wrapper' feature selection techniques: exhaustive feature selection, hill climbing and forward sequential selection. Then, the three parameters of ABE method are optimized by empirical trial and the optimal MIABE is compared to published results.

Comparisons of Feature Selection Techniques

Table 3.1 presents the results of each method's performance on three data splits with Euclidean distance, *K* = 3 and inverse distance weighted mean

(Kirsopp *et al.* 2002). The EX, HC, FSS and MI denote the EXhaustive feature selection, Hill Climbing feature selection, Forward Sequential feature selection, and MIABE respectively. The results show that MI achieves better testing performances under MMRE, PRED(0.25), and MdMRE than other wrapper methods while MI's training errors are among the largest ones. These findings confirm the argument that wrappers usually can yield high fitting accuracy but low generalization to other conditions (Peng *et al.*, 2005).

Table 3.1: Comparisons of different feature selection schemes

Feature selection methods	Training			Testing		
	MMRE	PRED(0.25)	MdMRE	MMRE	PRED(0.25)	MdMRE
MI	0.77	0.29	0.45	0.41	0.23	0.35
EX	0.67	0.29	0.39	0.49	0.20	0.46
HC	0.67	0.29	0.39	0.49	0.20	0.46
FSS	0.67	0.32	0.34	0.45	0.23	0.41

Euclidean distance, $K = 3$, and inverse distance weighted mean adaptation.

As mentioned in Section 3.3.1, MMRE stands for the mean value of MREs, PRED(0.25) calculates the proportion of those MREs which are equal to or less than 25%, and MdMRE is the median value of the MREs. These error metrics reflect different aspects of the statistical characteristics of MRE values. To further analyze the MRE values from the testing dataset, we draw out the box plot of MRE values in Fig 3.3. The boxplots illustrate the median, the inter-quartile range, and the outliers. The MI based feature selection shows a lower median line, and a slightly smaller inter-quartile range of the MRE values than other feature selection methods. Table 3.2 summarizes the selected features from the training data splits. Results from different feature selection techniques are presented.

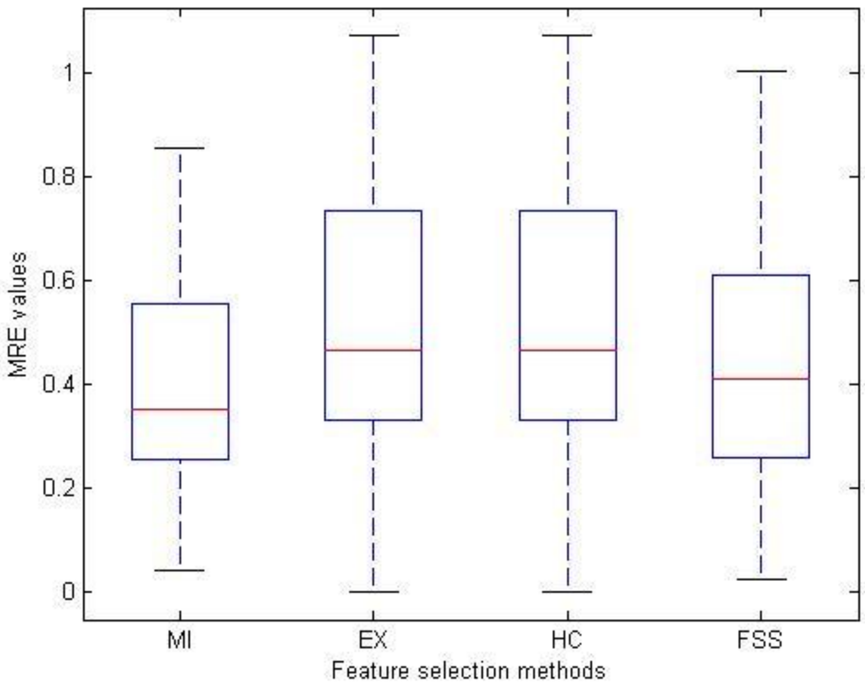

Figure 3.3: The boxplots of MRE values of feature selection methods

Table 3.2: Selected features in three data splits

Datasets / Variables	SP1				SP2				SP3			
	EX	FSS	HC	MI	EX	FSS	HC	MI	EX	FSS	HC	MI
TeamExp												
ManagerExp	1	1	1	1		1	1					1
YearEnd									1		1	
Transactions				1								1
Entities				1	1			1	1		1	1
PointsAdjust				1		1	1	1				1
Envergure				1	1			1				1
PointsNonAjust	1	1	1	1	1	1			1	1	1	1

The symbol '1' denotes the feature in its corresponding row is selected by the selection method in its corresponding column. The table shows that MI selects the same three features 'Entities', 'PointsAdjust', and 'Envergure' across all three splits. In particular, MI maintains the most informative feature, 'PointsAdjust'.

The mutual information diagram provides a useful graphic tool for a better understanding of the selected features. Fig 3.4 shows the mutual information diagram of the value of MI between different features and the cost value in the three data splits. The mutual information diagram provides useful information for diagnosing the feature extraction phase. In Fig 3.4, it is apparent that feature 7 ('PointsAdjust') has the highest amount of information shared with the cost value while feature 3 ('YearEnd') has the lowest mutual information with cost value. Surprisingly, EX and HC both select feature 3 in SP3 (see table 3.2). As the wrapper selectors use MMRE value on the training data for optimization and there is no clear relationship between mutual information and MMRE, a feature subset with low mutual information value may still achieve low MMRE value.

The computational complexity is another important criterion for evaluating feature selection method, especially when the dataset is large with more features and more projects, the speed of selection might have priority over accuracy. Therefore, the computational expense is also considered in our study. Time statistics needed for feature selection are provided in table 3.3.

The efficiency of each feature selection technique is measured by seconds. All methods are tested on a PC with Core Duo T2400 1.8GHz and 1G RAM. Table 3.3 shows that MI is most efficient among all methods.

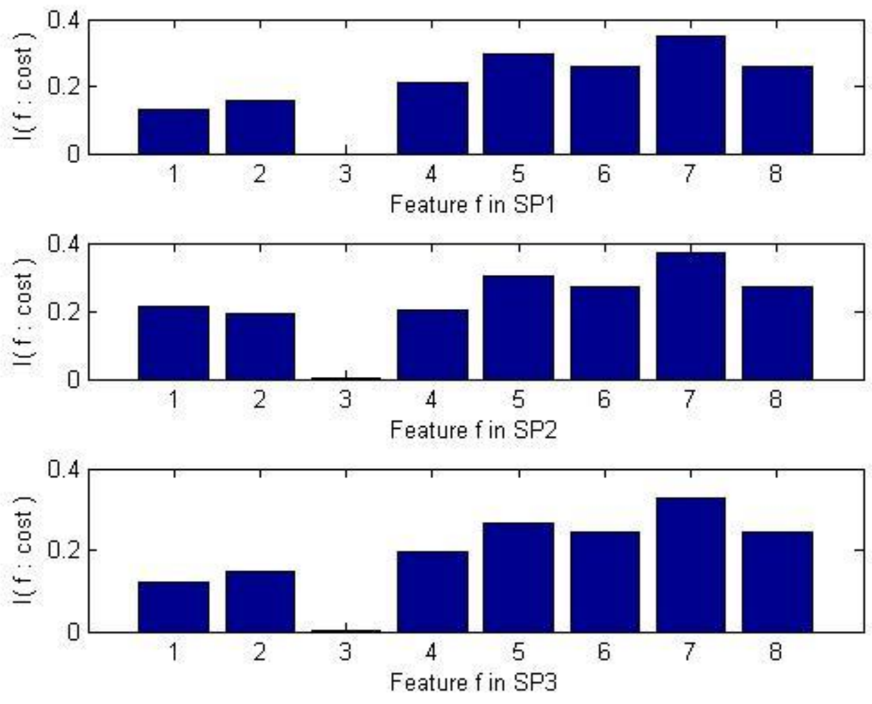

Figure 3.4: Mutual information diagram for the features in three training data splits

Table 3.3: Times consumed to optimize feature subset (seconds)

Data split	SP1	SP2	SP3
MI	3	2	3
EX	96	98	100
FSS	7	10	8
HC	47	45	39

Comparison of MIABE to Published Results

The three-fold cross-validation approach mentioned in section 3.3.3 is utilized to optimize ABE's parameters on Desharnais dataset. Table 3.4 summarizes the training and testing results with different parameter combinations from the parameter space: two distance measures (Euclidean distance and Manhattan distance), five K values (1, 2, 3, 4 and 5), and four solution functions (Closest Analogy (CA), Mean, Inverse distance Weighted Mean (IWM), and Median).

The results show that in general the choice of different similarity measures has an insignificant influence on both the training and testing performances. As for the solution functions, the closest analogy does not obtain best results and the median gets slightly better results than mean and IWM when $K = 4$ and $K = 5$. The choice of K value has some influence on the accuracies. Smaller errors are obtained when the estimation is based on a relatively larger number of analogies ($K = 4$, and $K = 5$). The best parameter combination (Manhattan similarity, $K = 4$, and median solution function) on the training dataset is selected to compare with the published training and testing results.

Table 3.4: MIABE estimation results on Desharnais Dataset

Similarity	*K* value	Solution function	Training MMRE	Training PRED(0.25)	Training MdMRE	Testing MMRE	Testing PRED(0.25)	Testing MdMRE
	K = 1	CA	0.83	0.24	0.48	0.50	0.27	0.44
	K = 2	Mean	0.77	0.29	0.47	0.50	0.27	0.42
		IWM	0.80	0.28	0.49	0.59	0.20	0.49
	K = 3	Mean	0.75	0.32	0.42	0.43	0.27	0.41
		IWM	0.77	0.29	0.45	0.41	0.23	0.35
		Median	0.75	0.29	0.42	0.41	0.27	0.35
Euclidean	K = 4	Mean	0.73	0.34	0.39	0.40	0.37	0.40
		IWM	0.76	0.28	0.43	0.40	0.43	0.28
		Median	0.71	0.34	0.44	0.40	0.33	0.39
	K = 5	Mean	0.71	0.32	0.43	0.44	0.33	0.38
		IWM	0.74	0.29	0.41	0.42	0.30	0.38
		Median	0.69	0.32	0.37	0.37	0.33	0.35
	K = 1	CA	0.78	0.24	0.47	0.42	0.37	0.36
	K = 2	Mean	0.78	0.30	0.42	0.46	0.23	0.39
		IWM	0.72	0.32	0.45	0.50	0.20	0.40
	K = 3	Mean	0.72	0.28	0.44	0.46	0.27	0.39
		IWM	0.74	0.30	0.41	0.48	0.27	0.40
		Median	0.76	0.30	0.44	0.42	0.33	0.32
Manhattan	K = 4	Mean	0.69	0.32	0.44	0.43	0.43	0.45
		IWM	0.71	0.30	0.43	0.44	0.33	0.44
		Median	0.68	0.32	0.39	0.36	0.40	0.33
	K = 5	Mean	0.71	0.33	0.39	0.41	0.40	0.36
		IWM	0.72	0.33	0.39	0.38	0.40	0.34
		Median	0.69	0.30	0.39	0.36	0.37	0.30

In table 3.5, the best MIABE (with Manhattan similarity, *K* = 4, and median solution function) is compared to the published results (Mair *et al.* 2000). In Mair *et al.* (2000) the statistics (Mean, Median, Min and Max) of the MRE values from three-folder cross-validation were given. Table 3.5 shows that MIABE obtains smallest mean, median, and min of MREs, and the second lowest max value of MREs. Although in our study the three-folder cross-validation is also used to build and test MIABE method, the conclusion that MIABE's results are better than the published results should be treated with caution. Because the three-folder cross-validation data splits are randomly generated, the splits used in our study may not be exactly the same as the ones used in Mair's study.

Table 3.5: Comparisons with published results

Published Techniques	MRE			
	Mean	Median	Min	Max
ANN (Mair et al. 2000)	0.47	0.53	0.21	0.66
ABE (Mair et al. 2000)	0.57	0.49	0.43	0.80
LSR (Mair et al. 2000)	0.62	0.47	0.38	1.00
RI (subset selection) (Mair et al. 2000)	0.90	0.89	0.41	1.41
MIABE	0.36	0.33	0.01	0.78

3.4.2 Results on Maxwell Dataset

The experimental results on Maxwell are presented in this section. In order to compare MIABE with published works, we prepare the training and testing datasets by following the splitting method of Maxwell's and Sentas *et al.*'s (Maxwell, 2002, Sentas *et al.* 2005). The 50 projects that completed before 1992 form the training set, and the 12 projects finished between 1992 and 1993 are used as testing set.

Comparison of Feature Selection Techniques

Table 3.6 summarizes the results of each selector's performance on training dataset and testing dataset with the same configuration used in

Desharnais dataset: Euclidean similarity, $K = 3$ and inverse distance weighted mean (Kirsopp *et al.* 2002). Due to large number of features (25 features), the exhaustive search is not applicable with our computation resource. The results show that MI achieves better or equally good testing performances when compared to wrapper methods, and MI's training performance is improved compared with the results in table 3.1. Specifically, MI's training PRED(0.25) and MdMRE rank first, and its training MMRE ranks second.

Table 3.6: Comparisons of different feature selection schemes

Feature selection	Training			Testing		
methods	MMRE	PRED(0.25)	MdMRE	MMRE	PRED(0.25)	MdMRE
MI	0.55	0.40	0.34	0.42	0.42	0.29
EX	NA[a]	NA	NA	NA	NA	NA
HC	0.58	0.40	0.40	1.01	0.42	0.42
FSS	0.48	0.34	0.35	0.45	0.33	0.31

Euclidean distance, $K = 3$, and inverse distance weighted mean adaptation.

[a]: Not Applicable

To further analyze the MRE values from the testing dataset, we draw out the box plot of MRE values in fig 3.5. It is shown that MI has a slightly lower median line than HC and FSS. The MI's inter-quartile range is smaller than HC's but a bit larger than FSS's.

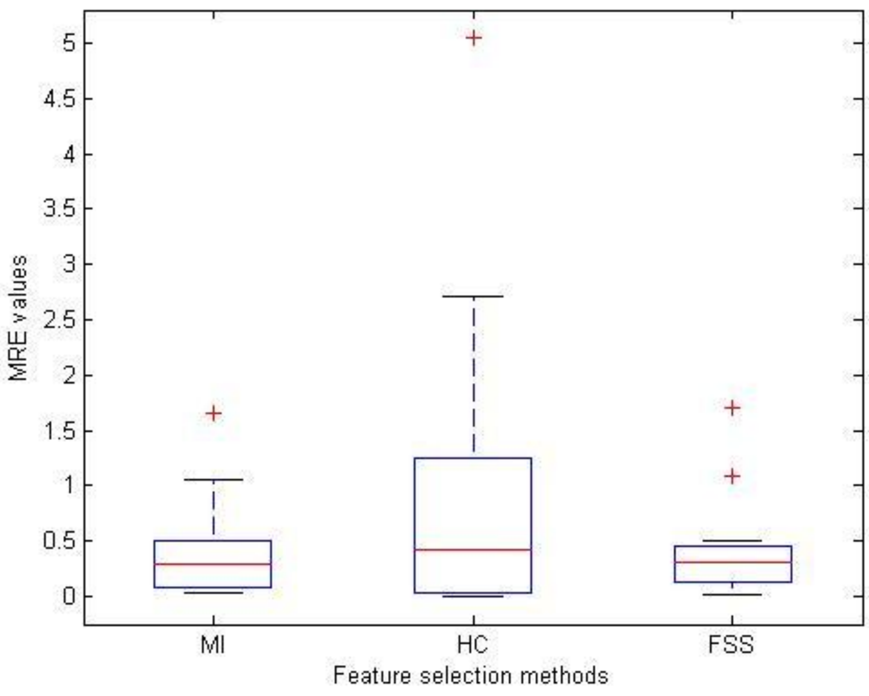

Figure 3.5: The boxplots of MRE values of feature selection methods (EX is not applicable)

Table 3.7 presents the selected features from the training dataset. The symbol ‘1’ denotes the feature in its corresponding row is selected by the selection method in its corresponding column. The symbol ‘-’ denotes the selection method in its corresponding column is not applicable. The table shows that MI selects three features ‘T14’, ‘Duration’ and ‘Size’, FSS chooses ‘T05’, ‘Duration’ and ‘Size’, and HC selects 12 out of 25 features.

Table 3.7: Selected variables for three splits

Datasets / Variables	Training Set			
	EX	FSS	HC	MI
Time	-		1	
App	-		1	
Har	-			
Dba				
Ifc	-			
Source	-		1	
Telonuse	-		1	
Nlan	-		1	
T01	-		1	
T02	-			
T03	-			
T04	-			
T05	-	1		
T06			1	
T07	-		1	
T08	-		1	
T09	-			
T10	-			
T11	-			
T12	-			
T13	-			
T14			1	1
T15				
Duration		1	1	1
Size		1	1	1

'1' denotes that the feature is not selected

Fig 3.6 shows the mutual information diagram of the value of MI between different features and the cost value in the training dataset. In fig 3.6, it is apparent that features 22, 24 and 25 ('T14', 'Duration' and 'Size') have the highest MIs, while features 5, 6 and 16 ('Ifc', 'Source', and 'T08') have the lowest MIs. Table 3.7 shows that features 22, 24 and 25 are selected by all methods, but HC selects feature 6 and 16.

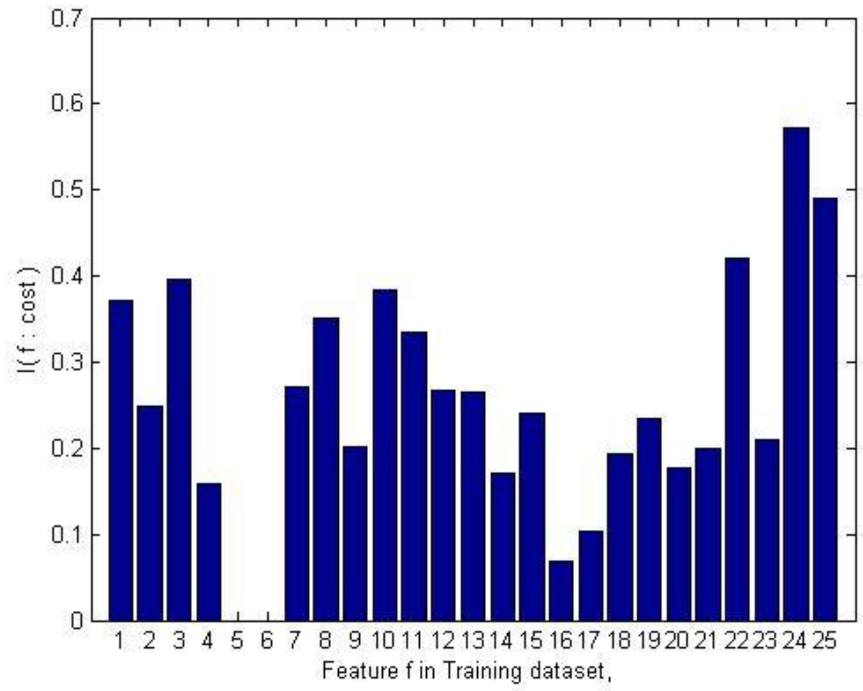

Figure 3.6: Mutual information diagram for the features in training dataset

The computation efficiency of each feature selector is also tested. The time (in seconds) consumed for selecting the optimal feature subset are provided in table 3.8 The results in Table 3.8 show that MI is fastest among all feature selection methods.

Table 3.8: Time needed to optimize feature subset (seconds)

Feature selection methods	Training set
MI	4
EX	NA*
FSS	31
HC	40

NA*: Not Applicable

Comparison of MIABE to Published Results

Similar to section 3.4.1, the parameters are optimized by trial-and-error scheme. The results of training datasets and testing datasets are presented in table 3.9. The results show that in general there is no clear conclusion on which similarity measure is better. As for the solution functions, the closest analogy does not obtain best results and the median and mean gets better testing results than IWM when $K = 4$ and $K = 5$. The choice of K value has small influence on the accuracies. The smaller errors are obtained when the estimation is based on a relatively larger number of nearest neighbors ($K = 4$, and $K = 5$). The best configuration (Euclidean similarity, $K = 4$, and mean solution function) on the training dataset is selected to compare with the published training and testing results.

In table 3.10, the best MIABE (with Euclidean similarity, $K = 4$, and mean solution function) is compared to the published results from stepwise regression (Maxwell 2002) and ordinal regression (Sentas *et al.* 2005). In their works, the MMRE and PRED(0.25) of training and testing datasets were given. Table 3.10 shows that MIABE obtains best training PRED(0.25) and testing PRED(0.25), and MIABE ranks third on both training and testing MMRE.

Table 3.9: MIABE estimation results on Maxwell Dataset

Similarity	*K* value	Solution function	Training MMRE	Training PRED(0.25)	Training MdMRE	Testing MMRE	Testing PRED(0.25)	Testing MdMRE
Euclidean	K = 1	CA	0.56	0.34	0.37	0.60	0.33	0.40
	K = 2	Mean	0.57	0.28	0.47	0.79	0.25	0.52
		IWM	0.55	0.24	0.44	0.97	0.42	0.67
	K = 3	Mean	0.56	0.38	0.35	0.37	0.50	0.27
		IWM	0.55	0.40	0.34	0.42	0.42	0.29
		Median	0.55	0.34	0.36	0.38	0.42	0.34
	K = 4	Mean	0.51	0.48	0.29	0.28	0.67	0.19
		IWM	0.55	0.40	0.41	0.64	0.17	0.42
		Median	0.50	0.34	0.37	0.27	0.58	0.22
	K = 5	Mean	0.58	0.32	0.38	0.37	0.58	0.22
		IWM	0.54	0.34	0.36	0.51	0.25	0.42
		Median	0.51	0.38	0.35	0.34	0.58	0.23
Manhattan	K = 1	CA	0.59	0.18	0.48	0.65	0.25	0.46
	K = 2	Mean	0.56	0.38	0.34	0.46	0.33	0.31
		IWM	0.55	0.40	0.34	0.52	0.25	0.33
	K = 3	Mean	0.54	0.34	0.32	0.39	0.50	0.25
		IWM	0.51	0.44	0.32	0.41	0.42	0.27
		Median	0.53	0.32	0.33	0.51	0.33	0.37
	K = 4	Mean	0.52	0.46	0.32	0.36	0.67	0.19
		IWM	0.51	0.36	0.35	0.51	0.50	0.32
		Median	0.50	0.34	0.43	0.34	0.58	0.22
	K = 5	Mean	0.53	0.30	0.35	0.37	0.58	0.20
		IWM	0.52	0.36	0.38	0.42	0.33	0.28
		Median	0.53	0.34	0.45	0.34	0.58	0.23

Table 3.10: Comparisons with published results

Published Techniques	Training MMRE	Training PRED(0.25)	Testing MMRE	Testing PRED(0.25)
Stepwise Regression (A) (Maxwell 2002)	0.42	0.42	0.29	0.33
Stepwise Regression (B) (Maxwell 2002)	0.43	0.34	0.32	0.58
Ordinal Regression (mean) (Sentas *et al.* 2005)	0.60	0.34	0.26	0.58
Ordinal Regression (median) (Sentas *et al.* 2005)	0.52	0.36	0.27	0.50
MIABE	0.51	0.48	0.28	0.67

3.4 Summary and Conclusion Remarks

Feature selection is a critical preprocessing stage of analogy based estimation. However, most existing feature selection methods for analogy based estimation are 'wrappers' (Kohavi and John, 1997). Usually, wrappers can yield high fitting accuracy at the cost of high computational complexity and low generalization of the selected features to other conditions. Another kind of feature selector 'filters' (Almuallim and Dietterich, 1994, Kohavi and John, 1997) has much lower complexity than wrappers and could select the features with interpretations, although 'filters' may not produce the fitting results as accurate as 'wrappers'. In our study, a novel algorithm that hybridizes wrapper and filter feature selection (MIABE) was proposed. The mutual information is used as the selection criterion for filter mechanism.

To validate the proposed MIABE algorithm, the experiments are conducted on two real world datasets. The performances of MIABE are first investigated by fixing the ABE parameter settings. It is compared against other wrapper feature selection methods (exhaustive search, hill climbing and forward sequential selection) for analogy based estimation. The prediction results suggested that MIABE could achieve better predictions on testing datasets (generalization) even though MIABE did not perform very well on fitting the training datasets.

In addition, the selected features by MIABE are analyzed by mutual information diagrams. The MI diagram provides useful information for diagnosing the feature selection phase. The results show that the MIABE can obtain more meaningful features which can be explained by MI diagram, while wrapper selectors do not always select informative features since they merely optimize the error metric MMRE. This could probably explain why MIABE could achieve better results in testing dataset. Moreover, the results also suggest that the mutual information based feature selection may be a feasible alternative when the wrapper techniques are facing over-fitting problems.

Another important finding is that MI based feature selection is more efficient than the wrappers, especially when there are large number of features in the dataset. This finding confirms the argument that the primary advantage of filter is the speed and ability to scale to large datasets.

Lastly, the MIABE is optimized and compared with the published works. The optimization took into account three parameters: similarity measure (including Manhattan distance and Euclidean distance), the number of nearest neighbors K (K from 1 to 5) and solution functions (including closest analogy, mean, inverse distance weighted mean and median). The comparisons show that MIABE achieves comparable results both on training and testing performances.

Chapter 4 Project Selection by Genetic Algorithm[2]

To improve ABE's performance, many studies, such as the work in the previous chapter, propose different approaches to optimize the weights of the project features (feature selection can be regarded as a special case of feature weighting with the value {0, 1}) in its similarity function. However, the historical database of ABE often contains noisy or redundant information, which can lead to poor prediction accuracy, large memory requirement, and excessive computation cost. To alleviate these drawbacks, we propose in this chapter the genetic algorithm for project selection for ABE (PSABE) which can reduce the whole historical database into a small subset that consists only of representative projects. Moreover, PSABE is combined with the feature weighting scheme (FWPSABE) for a further improvement. The proposed methods are validated on four datasets (two real-world datasets and two artificial datasets). The promising results indicate that the project selection technique could significantly improve ABE's prediction performance.

[2] The chapter is associated with the publications Li et al. 2007 and Li et al. 2009b

4.1 Introduction

A large number of research works have been focusing on the improvements of feature weighting/selection approaches, such as Shepperd and Schofield (1997), Walkerden and Jeffery (1999), Angelis and Stamelos (2000), Mendes *et al.* (2003), Auer *et al.* (2006), Huang and Chiu (2006), and Li *et al.* (2009a).

However, the historical database of ABE still confronts some difficulties, such as the non-normal characteristics including skewness, heteroscedasticity and excessive outliers (Pickard *et al.* 2001) and the ever increasing sizes of the datasets (Shepperd and Kadoda 2001). The large and non-normal historical databases always lead ABE methods to low prediction accuracy along with high computational cost (Huang *et al.* 2002). To alleviate these drawbacks, the project selection methodology has been proposed by some authors (Kirsopp and Shepperd, 2002). The objective of project selection (PS) is to identify and remove redundant and noisy projects. By reducing the whole project base into a smaller subset that consists only of the representative projects, project selection could save the computing time used for searching similar projects and produce quality prediction results.

Kirsopp and Shepperd (2002) first conducted a preliminary study on project selection using hill climbing, and forward and backward sequential selection. The combination of feature selection and project selection was also considered by Kirsopp and Shepperd (2002). However, they provide no clear conclusions that project selection could significantly improve ABE’s accuracy. This might be due to the fact that the optimization algorithms they used are not powerful enough to achieve global optimum and the feature selection scheme is limited to the space {0, 1} with two elements. On the contrary, the feature weighting scheme has a much larger space: [0, 1].

In this study, we propose genetic algorithm (GA) to perform the optimization task. GA is a robust global optimization technique which usually converges rapidly to solutions of good quality. Moreover, GA is capable of optimizing the continuous feature weights which is an extension of the feature selection problem. It is difficult for the heuristics like forward sequential selection to optimize continuous variables. Additionally, in CBR literature it has been frequently reported that the simultaneous optimization of feature weighting and project selection by GA can significantly improve CBR’s prediction accuracy (Kuncheva and Jain 1999, Rozsypal and Kubat 2003, Ahn *et al.* 2006).

Therefore, it is worthwhile to investigate GA for project selection. In this chapter, we propose GA for project selection for ABE (PSABE) and the simultaneous optimization of feature weights and project selection for ABE (FWPSABE). The proposed two techniques are compared against the feature weighting ABE (ABE), the conventional ABE and other popular cost estimation methods including ANN, RBF, SVM and CART.

The rest of this chapter is organized as follows: section 4.2 presents the general framework of feature weight and project selection system for ABE. Section 4.3 presents the real world datasets and the experiments design. In section 4.4, the results on two real world data sets are summarized and analyzed. In section 4.5, two artificial data sets are generated, experiments are conducted on two artificially generated datasets, and the results are summarized and analyzed.

4.2 Project Selection and Feature Weighting

In this section, we construct the FWPSABE system which can perform FWABE, PSABE, and simultaneous Feature Weights and Project Selection Analogy Based Estimation (FWPSABE). Genetic algorithm (Holland 1975) is selected as the optimization tool for the FWPSABE system, as it is a robust global optimization technique and has been applied to optimize the model parameters by several cost estimation papers (Dolado 2000, Shukla 2000, Dolado 2001, Huang and Chiu 2006). The framework and detailed description of FWPSABE system are presented in the following paragraphs.

The system procedures of project selection and feature weighting via GA are given in this section. The system consists of two stages: the first stage is the supervised training stage (as shown in fig. 4.2) and the second stage is the testing stage (as shown in fig. 4.3). In the training stage, a set of training projects is presented to the system, the ABE method is configured by the candidate parameters (feature weights and selection codes) to produce the cost predictions, and GA explores the parameter space to minimize the error (in terms of MMRE) of ABE on the training projects by the following steps:

i. Encoding.

To apply GA for optimization, the candidate parameters are coded as a binary code chromosome. As shown in fig 4.1, each individual chromosome consists of two parts. The first part is the codes for feature weights with a length of $14 \times n$, where n is the number of features. Since the feature weights in the ABE model are decimal numbers, the binary codes have to be transformed into decimal values before entering the ABE method. As what many authors (Michalewicz 1996, Ahn *et al.* 2006) have suggested, the features weights are set as precisely as 1/10,000. Thus, 14 binary bits are required to express this precision level because $8192 = 2^{13} < 10000 \leq 2^{14} = 16384$. After the transformation, all decimal weight values are normalized into the interval [0, 1] by the following formula (Michalewicz 1996):

$$w_i = \frac{w_i'}{2^{14} - 1} = \frac{w_i'}{16383} \tag{4.1}$$

where w_i' is the decimal conversion of the ith feature's binary weight. For example, the binary code for the first feature of the sample chromosome in fig 4.1 is $(10000000000001)_2$. Its decimal value is $(8193)_{10}$ and its normalized value is 8193/16383 ≈ 0.5001.

The second part of the codes is for project selection. The value of each bit is set to be either 0 or 1: 0 means the corresponding project is not selected and 1 means it is selected. The length of second part is m, and m is the total number of projects in the historical project base.

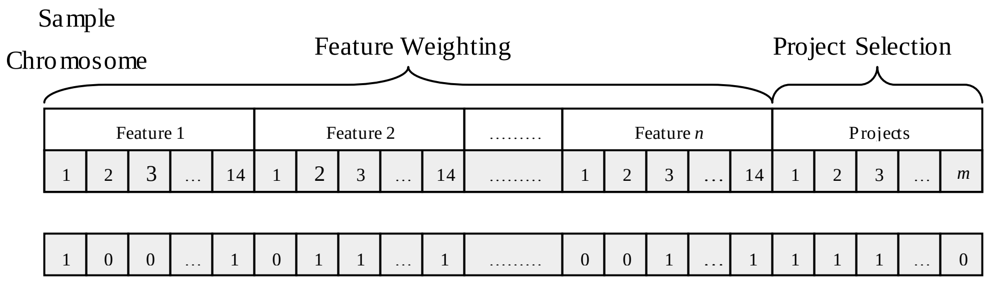

Figure 4.1: Chromosome for FWPSABE

ii. Population generation.

After the encoding of the individual chromosome, the algorithm generates a population of chromosomes with an initialization probability of 0.5 (It means that each bit in the population has an equal chance to be ‘1’ or ‘0’). For the GA process, larger population size often results in higher chances for optimal solutions (Doval *et al.* 1999). Since GA is computationally expensive, a trade-off between the convergence time and the population size must be made. In general, the minimum effective population size grows with problem size. Based on some previous works (Huang and Chiu 2006, Chiu and Huang 2007), the size of the population is set to be 10V where V is the total number of input variables of GA search, which partially reflects the problem size.

iii. Fitness function.

Each individual chromosome is evaluated by the fitness function in GA. *MMRE* is chosen to establish the fitness function but GA is designed to maximize the fitness value. For simplicity, we set the fitness function as the reciprocal of MMRE plus a small constant $\delta = 0.001$ which is used to prevent the situation that MMRE = 0.

$$f = \frac{1}{MMRE + \delta} \tag{4.2}$$

iv. Fitness evaluation.

After transforming the binary chromosomes into the feature weighting and project selection parameters (see step *i*), the procedures of ABE are executed as follows:

- Given one training project, the similarities between the training project and historical projects are computed by assigning the feature weights into the similarity functions.
- Simultaneously, the project selection part of the chromosome is used to generate the reduced historical project databases (Reduced PDs).
- Then, ABE uses 1 to 5 nearest neighbors (*K* = 1 to 5) matching to search through the reduced PD for 1 to 5 most similar historical projects.
- Finally, the ABE model assigns a prediction value to the training project by adopting one solution function.

The error metrics MMRE, PRED(0.25), and MdMRE are used to evaluate the prediction performance on the training project set. Then, the fitness value in step *iii* is calculated for each parameter combination (or chromosome).

v. Selection.

The standard roulette wheel mechanism is applied to select a number of $10V$ chromosomes from the current population.

vi. Crossover.

The selected chromosomes are consecutively paired. The 1-point crossover operator with a probability of 0.7 is used to produce new chromosomes. The newly created chromosomes constitute a new population.

vii. Mutation.

After crossover operation, each bit of the chromosomes in the new population is chosen to change its value with a probability of 0.1, in such a way that a bit ‘1’ is changed to ‘0’ and a bit ‘0’ is changed to ‘1’.

viii. Elitist strategy.

Elitist strategy is used to overcome the defect of the slow convergence rate of GA. The elitist strategy retains good chromosomes and ensures they are not eliminated through the mechanism of crossover and mutation. Under this strategy, if the minimum fitness value of the new population is smaller than that of the old population, then the new chromosome with the minimum fitness value will be replaced with the old chromosome with the maximum fitness value.

viii. Stopping criteria.

There are few theoretical guidelines for determining when to terminate the genetic search. By following the previous works (Huang and Chiu 2006, Chiu and Huang 2007) on GA combined with the ABE method, steps *v* to *viii* are repeated until the number of generations is equal to or exceeds 1000*V* trials or the best fitness value does not change in the past 100*V* trials. After the stopping criteria are satisfied, the system moves to the second stage and the optimal parameters or chromosome are entered into the ABE model for testing.

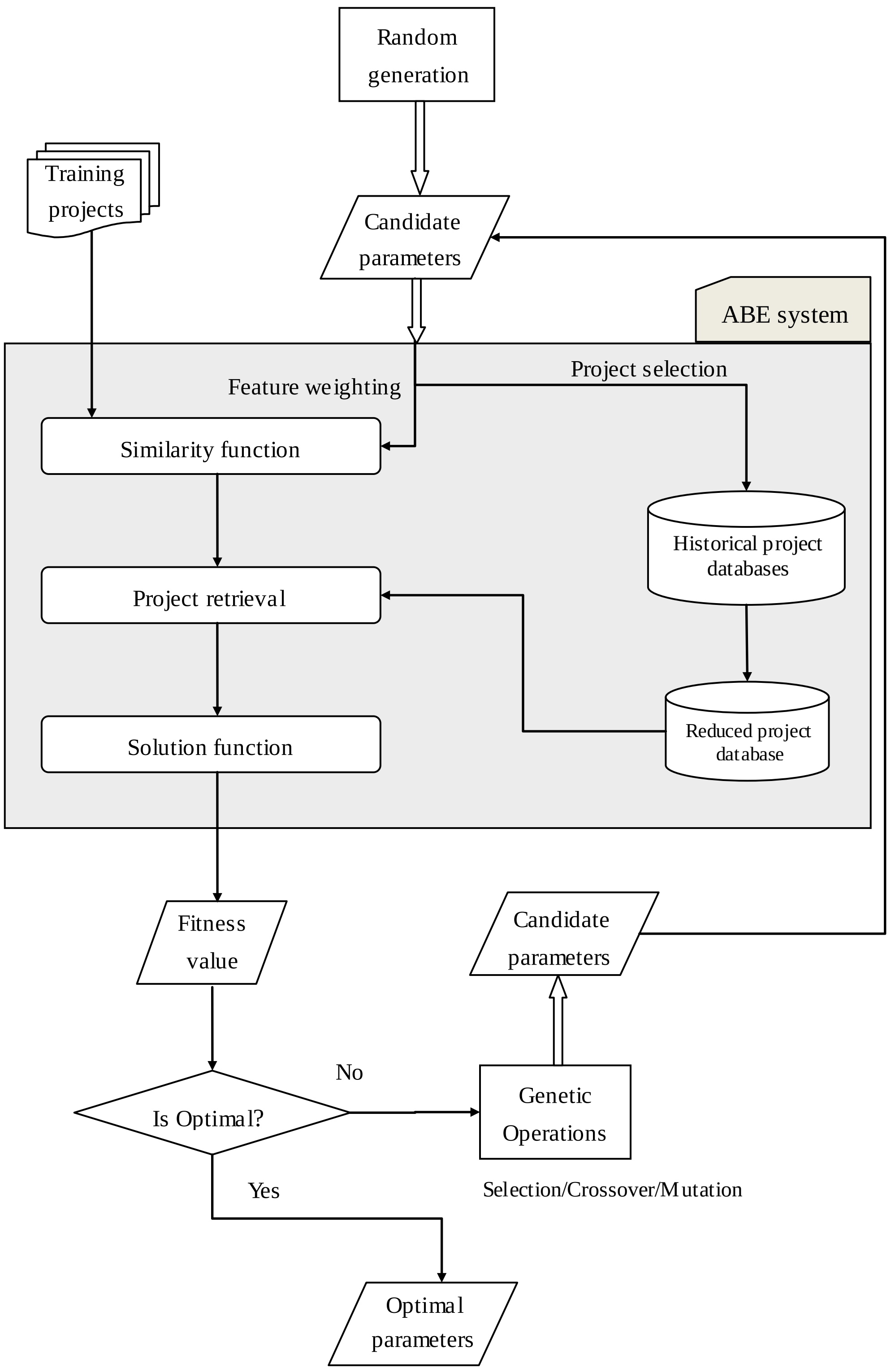

Figure 4.2: The training stage of FWPSABE

In the above procedure, the population size, crossover rate, mutation rate and stopping condition are the controlling parameters of the GA search. However, there are few theories to guide the assignments of these values (Ahn *et al.* 2006). Hence, we determine the value of these parameters in the light of previous studies that combines ABE and GAs. Most prior studies use 10*V* chromosomes as the population size, and their crossover rate ranges from 0.5 to 0.7, while the mutation rate ranges from 0.06 to 0.1 (Ahn *et al.* 2006, Huang and Chiu 2006, Chiu and Huang 2007). However, because the search space for our GA is larger than these studies (the number of input variables *V* is larger than that in previous papers), we set the parameters to the higher bounds of those ranges. Thus, in this study the population size is 10*V*, the crossover rate is set at 0.7 and the mutation rate is set at 0.1.

The second stage is the testing stage. In this stage, the system receives the optimized parameters from the training stage to configure the ABE model. The optimal ABE is then applied to the testing projects to evaluate the trained ABE.

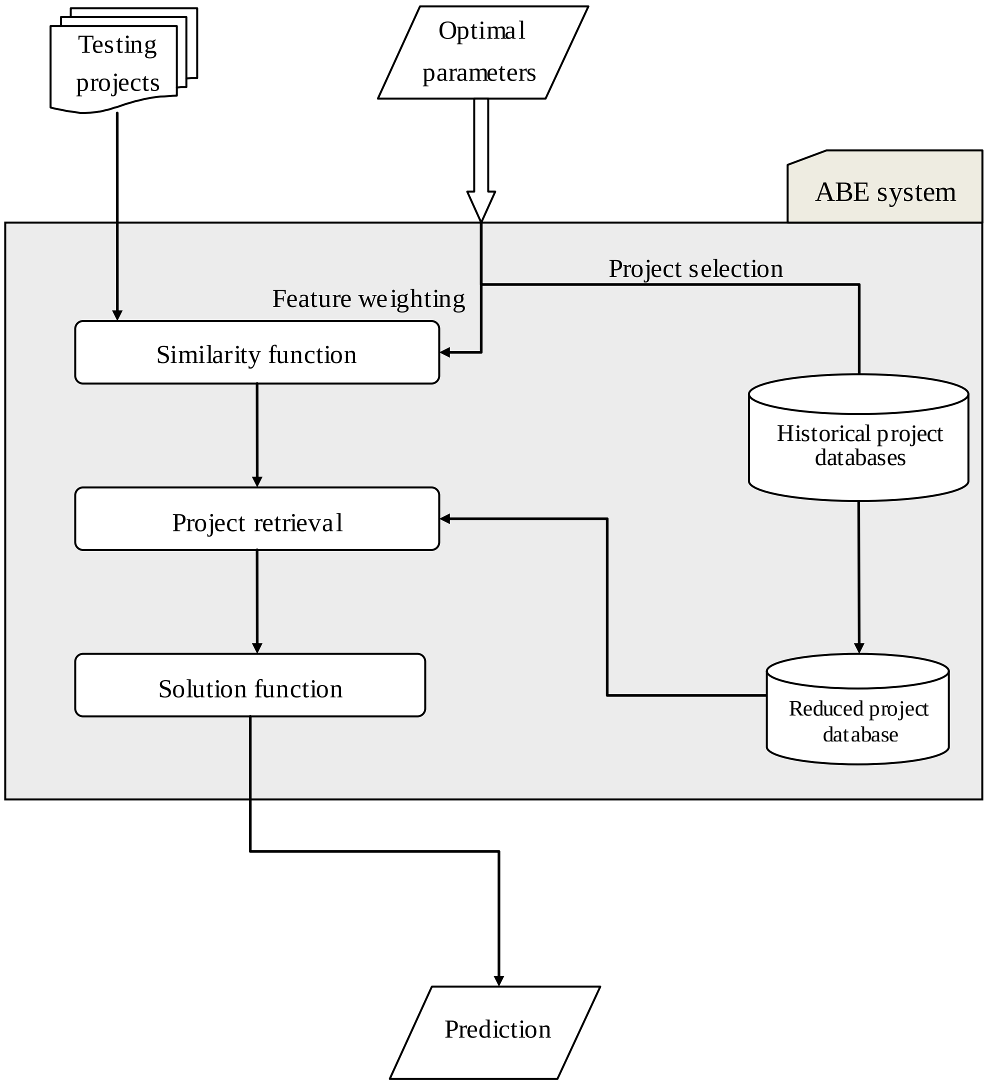

Figure 4.3: The testing stage of FWPSABE

4.3 Experiment Design

In this section, two real world software engineering datasets are used for the experiments and the detailed experiments designs are presented.

4.3.1 Datasets

The Albrecht dataset (Albrecht and Gaffney 1983) includes 24 projects developed by using third generation languages. 18 of the projects were written in COBOL, 4 were written in PL1, and 2 were written in DMS languages. Six independent features of this data set are ‘input count’, ‘output count’, ‘query count’, ‘file count’, ‘function points’, and ‘lines of source code’. The dependent feature ‘person hours’ is recorded in 1000 *h*. The detailed definitions and descriptive statistics of all features are shown in table B.1 and table B.2 of Appendix B.

The Desharnais dataset (Desharnais 1989) includes 81 projects and 11 features, 10 independent and one dependent. Since 4 out of 81 projects contain missing feature values, they have been excluded from the dataset. This process results in the 77 complete projects for our study. The ten independent features of this dataset are ‘TeamExp’, ‘ManagerExp’, ‘YearEnd’, ‘Length’, ‘Transactions’, ‘Entities’, ‘PointsAdjust’, ‘Envergure’, ‘PointsNonAjust’, and ‘Language’. The dependent feature ’person hours’ is recorded in 1000 *h*. The detailed definitions and descriptive statistics of all the features are shown in table B.3 and table B.4 of Appendix B.

4.3.2 Experiment Design

Before the experiments, all types of features are normalized into the interval [0, 1] in order to eliminate their different influences. The three-fold cross-validation is used to assess the accuracies of our method, similarly to Briand *et al.* (1999), Jeffery *et al.* (2001), and Mendes *et al.* (2003). Under this scheme, the data set is randomly split into 3 equally sized subsets. At each time, one of the three subsets is used as the testing set exclusively for evaluating model prediction, and the remaining two subsets are merged to form a training set which is only used to construct the models. This process is repeated three times and each subset has been used for testing only once. Finally the average training error and testing error across all three trials are computed.

Methods specifications

Four ABE based models are included in our experiments: conventional ABE, FWABE (feature weighting analogy based estimation) (Huang and Chiu 2006), PSABE (project selection analogy based estimation), and FWPSABE (simultaneous optimization of features weighting and project selection).

For a comprehensive evaluation of the proposed models, we include other popular machine learning methods including artificial neural network (ANN) (Heiat 2002), radial basis functions (RBF) (Shin and Goel 2000), support vector machine regression (SVR) (Oliveira 2006), and classification and regression trees (CART) (Pickard *et al.* 2001). The best variants of machine learning methods are obtained by training these methods and tuning their parameters on the historical datasets and training datasets respectively.

In ANN model, the number of hidden layers, the number of hidden nodes and the type of transfer functions are three predefined parameters. They have significant impacts on the prediction performance (Martin *et al.* 1997). Among these parameters, one hidden layer is often recommended since multiple hidden layers may lead to an over parameterized ANN structure with over-fitting characteristic. Thus, in this study we fix the number of hidden layers at 1. The search spaces for the number of hidden neurons and hidden layer transfer functions are set to be {1, 3, 5, 7, 9, 10} and {linear, tan-sigmoid, log-sigmoid} respectively. During the training process, the ANN models with different parameter configurations are first trained on the historical dataset. Then, all ANN versions are implemented on the training set and the one producing the lowest MMRE value is selected for the comparisons against ABE models.

For RBF network, the forward selection strategy is utilized since forward selection has the advantages of flexible number of hidden nodes, the tractable model selection criteria and the relatively low computational expense (Orr 1996). In this case, the regularization parameter λ is introduced. To determine λ, its search space is defined as $\lambda = \{10^i \mid i = -10, -9, ..., 0, ..., 10\}$. Similar to ANN's training procedure, all RBFs with different λ values are trained on the historical dataset and the one yielding the lowest MMRE on training data is selected for comparisons.

For SVR model, the common Gaussian function $K(x, y) = \exp\{-(x-y)^2 \delta^2\}$ is used as the kernel function. The predefined parameters δ, C and ε, are selected from the same search space $\{10^i \mid i = -10, -9, ..., 0, ..., 10\}$. SVR models with all kinds of parameters combinations ($10 \times 10 \times 10 = 1000$ combinations) are trained on the historical dataset. The combination producing the minimal MMRE on the training set is chosen for comparisons.

To train CART model, we first use the historical set to fit the model and obtain a decision tree *T*. The tree *T* is then applied to the training set, and returns a vector of cost values computed for the training projects. The cost vector is subsequently used to prune the tree *T* into a size that is minimized. The tree with optimal size is adopted for comparisons.

Experiment procedure

The following experiment procedures are conducted for comparisons:

- Firstly, the performances of FWPSABE are investigated by varying ABE parameters other than feature weights and project subsets. In line with the common settings of ABE parameters, we define the search spaces for similarity function as {Euclidean similarity, Manhattan similarity}, number of nearest neighbors *K* as {1, 2, 3, 4, 5}, and solution functions as {closest analogy, mean, median, inverse distance weighted mean} respectively. All kinds of parameter combinations are executed on both the training dataset and the testing dataset. The best configuration on training dataset is selected out for the comparisons with other cost estimation methods.
- Secondly, other ABE based methods are trained by the similar procedure

described in the first step and the best variants on training set are selected as the candidates for comparisons. In addition, the optimizations of machine learning methods are conducted on the training dataset by searching through their parameter spaces.

- Thirdly, the training and testing results of the best variants of all estimation methods are analyzed and compared. The experiments results and analyses are presented in the next section.

4.4 Results

4.4.1 Results on Albrecht Dataset

Table 4.1 presents FWPSABE's results on Albrecht dataset with different parameter configurations mentioned in section 4.3. The results show that in general, Euclidean similarity achieves slightly more accurate performances than Manhattan similarity on both the training and testing datasets. As for the solution function, there is no clear observation on which function is most preferable. The choice of K value has some influence on the accuracies. The smaller errors mostly appear when $K = 3$ and $K = 4$. Among all configurations, the setting {Euclidean similarity, $K = 4$, and mean solution function} produces best results on the training dataset and so it is selected for the comparisons with other cost estimation methods.

Table 4.2 summarizes the results of the best variants of all cost estimation methods on Albrecht dataset. It is observed that the FWPSABE achieves the best testing performance (0.30 for MMRE, 0.63 for PRED(0.25) and 0.27 for MdMRE) among all methods, followed by PSABE, and FWABE. For a better illustration, the corresponding testing performances are presented in fig 4.4.

Table 4.1: Results of FWPSABE on Albrecht Dataset

Similarity	*K* value	Solution	Training			Testing		
			MMRE	*PRED*(0.25)	*MdMRE*	*MMRE*	*PRED*(0.25)	*MdMRE*
Euclidean	K = 1	CA	0.39	0.25	0.35	0.40	0.38	0.45
	K = 2	Mean	0.37	0.54	0.34	0.55	0.13	0.58
		IWM	0.40	0.58	0.34	0.57	0.32	0.42
	K = 3	Mean	0.56	0.38	0.34	0.41	0.33	0.39
		IWM	0.55	0.42	0.32	0.42	0.42	0.29
		Median	0.55	0.38	0.33	0.38	0.46	0.32
	K = 4	Mean	0.31	0.54	0.32	0.30	0.63	0.27
		IWM	0.35	0.52	0.33	0.44	0.50	0.32
		Median	0.40	0.54	0.37	0.37	0.58	0.28
	K = 5	Mean	0.58	0.42	0.32	0.39	0.38	0.45
		IWM	0.54	0.33	0.38	0.51	0.25	0.42
		Median	0.51	0.38	0.45	0.42	0.25	0.45
Manhattan	K = 1	CA	0.50	0.25	0.41	0.45	0.25	0.53
	K = 2	Mean	0.56	0.38	0.42	0.43	0.13	0.44
		IWM	0.55	0.40	0.44	0.59	0.28	0.45
	K = 3	Mean	0.55	0.52	0.45	0.39	0.38	0.35
		IWM	0.51	0.44	0.42	0.42	0.25	0.40
		Median	0.53	0.32	0.43	0.51	0.33	0.32
	K = 4	Mean	0.53	0.38	0.32	0.41	0.54	0.45
		IWM	0.51	0.36	0.35	0.51	0.50	0.42
		Median	0.50	0.34	0.43	0.44	0.53	0.32
	K = 5	Mean	0.54	0.34	0.42	0.59	0.13	0.58
		IWM	0.52	0.36	0.48	0.52	0.23	0.48
		Median	0.53	0.34	0.45	0.51	0.13	0.46

Table 4.2: The results and comparisons on Albrecht Dataset

Models	*MMRE*		*PRED(0.25)*		*MdMRE*	
	Training	Testing	Training	Testing	Training	Testing
ABE	0.38	0.49	0.50	0.13	0.36	0.49
FWABE	0.48	0.42	0.38	0.25	0.34	0.46
PSABE	0.40	0.39	0.25	0.38	0.35	0.45
FWPSABE	0.31	0.30	0.54	0.63	0.32	0.27
SVR	0.46	0.45	0.50	0.25	0.22	0.43
ANN	0.39	0.49	0.38	0.25	0.35	0.51
RBF	0.79	0.49	0.50	0.25	0.25	0.39
CART	4.77	1.70	0.13	0.13	0.58	0.89

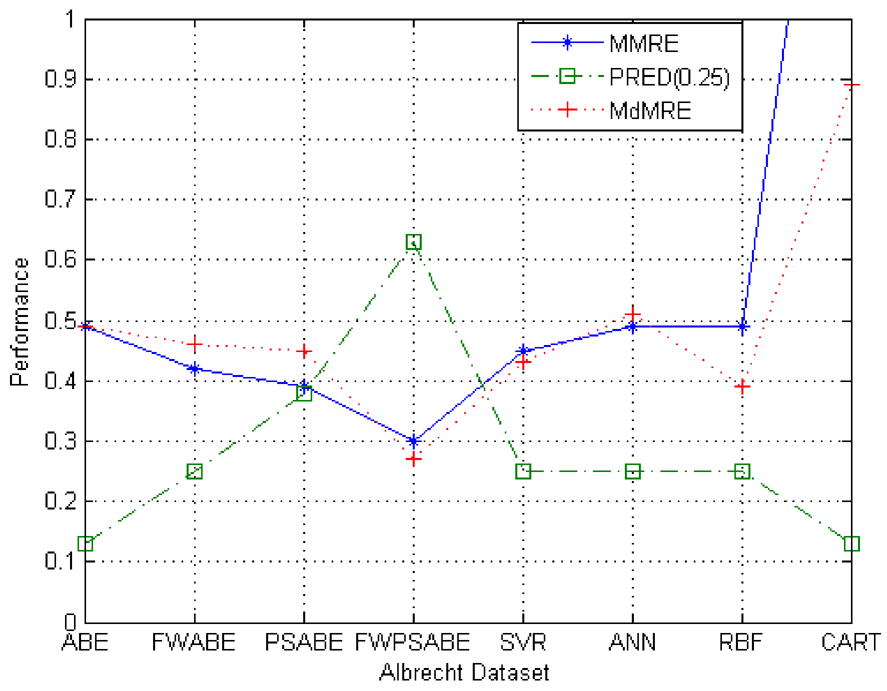

Figure 4.4: The testing results on Albrecht Dataset

4.4.2 Results on Desharnais Dataset

The results of FWPSABE with different configurations on Desharnais dataset are summarized in table 4.3. The results show that in this case the choice of different similarity functions has little influence on both the training and testing performances. As for the solution functions, there is no clear conclusion on which solution function is the best. The choice of *K* value has slight influence on the accuracies. The smaller errors are achieved by setting *K* = 3. In all configurations, the setting {Euclidean similarity, *K* = 3, and mean solution function} produces best results on the training dataset and so it is selected for the comparisons against other cost estimation methods.

Table 4.4 presents the results of the best variants of all cost estimation methods on Desharnais dataset. It is shown that the FWPSABE achieves the best testing performance (0.32 for MMRE, 0.44 for PRED(0.25) and 0.29 for MdMRE), and followed by SVR and PSABE. Fig 4.5 provides an illustrative version of the testing results in Table 4.4.

Table 4.3: Results of FWPSABE on Desharnais Dataset

Similarity	*K* value	Solution	Training			Testing		
			MMRE	*PRED*(0.25)	*MdMRE*	*MMRE*	*PRED*(0.25)	*MdMRE*
Euclidean	K = 1	CA	0.54	0.24	0.47	0.52	0.27	0.51
	K = 2	Mean	0.57	0.26	0.45	0.62	0.37	0.50
		IWM	0.55	0.24	0.44	0.97	0.42	0.67
	K = 3	Mean	0.40	0.36	0.36	0.32	0.44	0.29
		IWM	0.55	0.36	0.38	0.42	0.42	0.36
		Median	0.56	0.34	0.36	0.38	0.42	0.34
	K = 4	Mean	0.59	0.16	0.39	0.40	0.26	0.39
		IWM	0.55	0.36	0.41	0.64	0.17	0.46
		Median	0.53	0.34	0.37	0.57	0.38	0.42
	K = 5	Mean	0.55	0.24	0.56	0.43	0.28	0.48
		IWM	0.54	0.26	0.56	0.52	0.25	0.42
		Median	0.59	0.29	0.55	0.64	0.27	0.53
Manhattan	K = 1	CA	0.39	0.28	0.37	0.67	0.30	0.44
	K = 2	Mean	0.54	0.32	0.48	0.47	0.25	0.51
		IWM	0.55	0.40	0.34	0.52	0.25	0.53
	K = 3	Mean	0.45	0.28	0.49	0.46	0.22	0.38
		IWM	0.56	0.24	0.43	0.41	0.42	0.37
		Median	0.58	0.20	0.46	0.51	0.20	0.45
	K = 4	Mean	0.51	0.24	0.48	0.57	0.33	0.51
		IWM	0.53	0.26	0.55	0.58	0.27	0.52
		Median	0.60	0.30	0.53	0.54	0.28	0.52
	K = 5	Mean	0.54	0.24	0.50	0.52	0.26	0.48
		IWM	0.56	0.34	0.58	0.64	0.18	0.59
		Median	0.63	0.36	0.55	0.55	0.23	0.52

Table 4.4: The results and comparisons on Desharnais Dataset

Models	*MMRE*		*PRED(0.25)*		*MdMRE*	
	Training	Testing	Training	Testing	Training	Testing
ABE	0.62	0.62	0.28	0.22	0.51	0.50
FWABE	0.51	0.46	0.12	0.22	0.48	0.39
PSABE	0.39	0.41	0.28	0.30	0.37	0.38
FWPSABE	0.40	0.32	0.36	0.44	0.36	0.29
SVR	0.42	0.40	0.28	0.37	0.45	0.37
ANN	0.45	0.57	0.36	0.22	0.44	0.43
RBF	0.57	0.42	0.24	0.37	0.49	0.29
CART	0.97	0.52	0.28	0.30	0.50	0.35

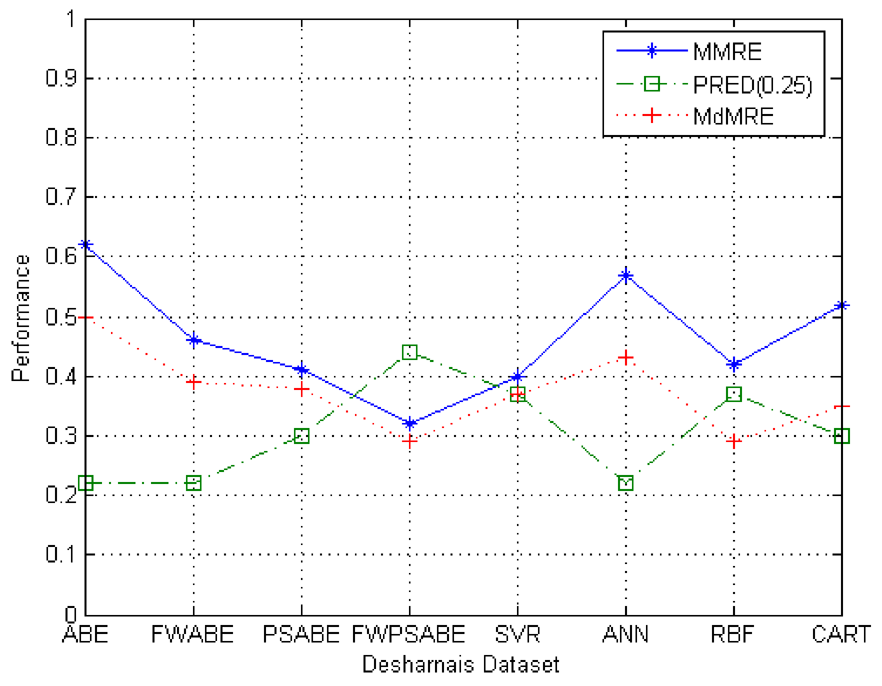

Figure 4.5: The testing results on Desharnais Dataset

4.5 Artificial Datasets and Experiments on Artificial Datasets

To compare different cost estimation methods, the empirical validation is very crucial when the theoretical comparisons are difficult to conduct. This has led to the collection of various real world data sets for experiments. Mair *et al.* (2005) conducted an extensive survey of the real data sets for cost estimation from 1980 onwards. As reported, most published real world data sets are relatively small for the tests of significance and their true properties may not be fully known. For example, it might be difficult to distinguish different types of distribution in the presence of extreme outliers in a small data set (Shepperd and Kadoda, 2001).

Artificially generated data sets provide a feasible solution to the above two difficulties. Firstly, the researchers can generate reasonable amount of artificial data to investigate the significant differences among the competing techniques. Secondly, it provides the control over the characteristics of the artificial dataset. Particularly, researchers could design a systematic way to vary artificial dataset properties for their research purposes (Pickard *et al.* 1999). In order to evaluate the proposed methods in a more convincing way, we generate two artificial datasets for further experiments.

From each of the two real data sets, we extract a set of characteristics describing its property, or more specifically its non-normality. The non-normality considered in our study includes skewness, variance instability, and excessive outliers (Pickard *et al.* 2001). We then use the two sets of characteristics to generate two sets of artificial data. Section 4.5.1 presents the details for artificial dataset generation.

4.5.1 Generation of Artificial Datasets

To explore the non-normal characteristics of the real world data set, the 'cost-size' scatter plot for Albrecht dataset is drawn in fig 4.6. The scatter plot indicates the slight skewness, moderate outliers, and slight variance instability of the Albrecht dataset. The 'cost-size' scatter plot of the Desharnais dataset is in fig 4.7, which shows heavier skewness, more extreme outliers, and higher variance instability.

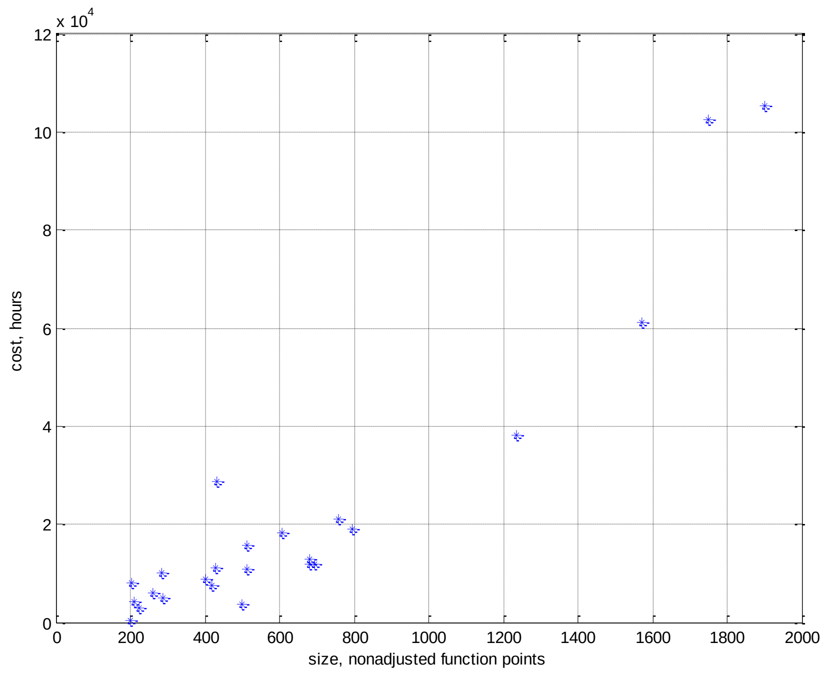

Figure 4.6: Cost versus size of Albrecht dataset

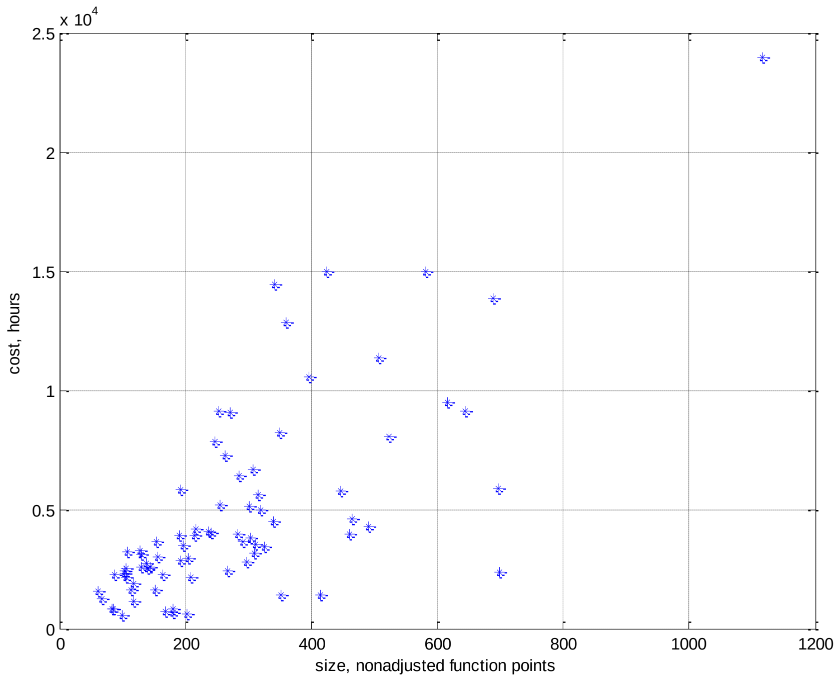

Figure 4.7: Cost versus size of Desharnais dataset

From the analysis above, software data sets often exhibit a mixture of several non-normal characteristics such as skewness, variance instability, and excessive outliers (Pickard *et al.* 2001). These characteristics do not always appear in the same degree. In some cases they are moderately non-normal such as the Albrecht dataset, while in other cases they are severely non-normal such as the Desharnais dataset. We adopted Pickard's method of non-normality modeling in this work. Other types of techniques for artificial dataset generation are also available in recent literature. For more details, readers can refer to Shepperd and Kadoda (2001), Foss *at al.* (2003), and Myrtveit *et al.* (2005).

Following Pickard's method, we simulate the combination of non-normal characteristics: skewness, unstable variance and outliers in formula (4.3):

$$y = 1000 + 6x_1sk + 3x_2sk + 2x_3sk + e_{h\ e} \qquad (4.3)$$

The independent variables (x_1sk, x_2sk, x_3sk) are generated by Gamma distributed random variables x_1', x_2', and x_3' with mean 4 and variance 8. The skewness is embedded in Gamma distributions. In order to vary the scale of the independent variables, we then multiply x_1' by 10 to create the variable x_1sk, x_2' by 3 to create the variable x_2sk and x_3' by 20 to create the variable x_3sk.

The last term e_{het} in formula (4.3) simulates a special form of unstable variance: heteroscedasticity. The heteroscedasticity occurs where the error term is related to one of the variables in the model and either increases or decreases depending on the value of the independent variable. The error term e_{het} is related to x_1sk via the formula $e_{het} = 0.1 \times e \times x_1sk$ for the moderate heteroscedasticity, and $e_{het} = 6 \times e \times x_1sk$ for the severe heteroscedasticity (Pickard *et al.* 2001).

The outliers are generated via multiplying or dividing the dependent variable y by a constant. We select 1% of the data to be the outliers. Half of the outliers are obtained by multiplying, while half of them are produced by dividing. For the moderate outliers, we set the multiplier/divider as 2, while for the severe outliers, we set the multiplier/divider to be 6.

The combination of moderate heteroscedasticity and moderate outliers is used to generate the moderate non-normality dataset (fig 4.8). The combination of severe heteroscedasticity and severe outliers is used to obtain the severe non-normality dataset (fig 4.9).

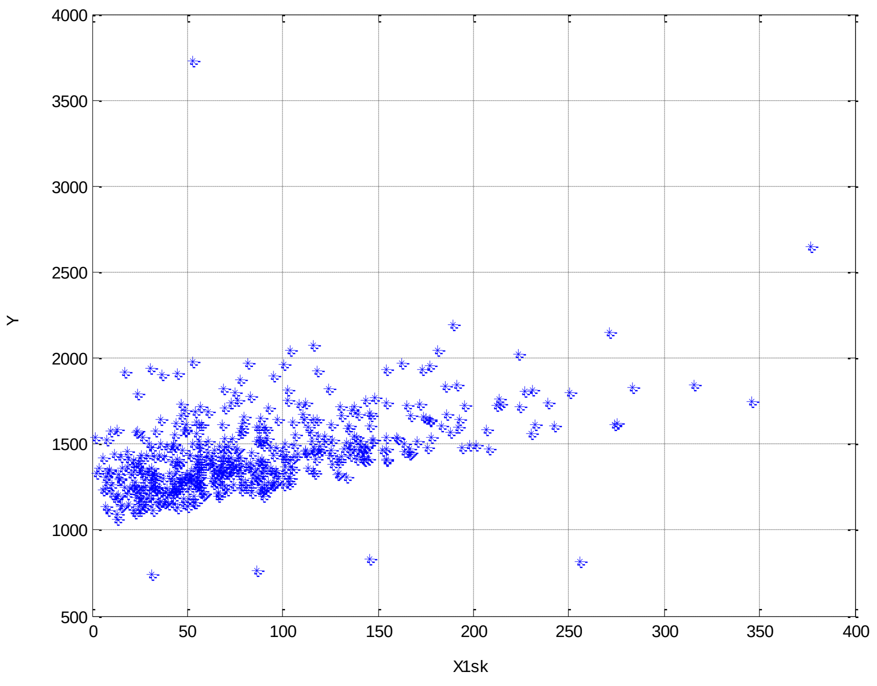

Figure 4.8: Y versus x_1sk of moderate non-Normality Data set

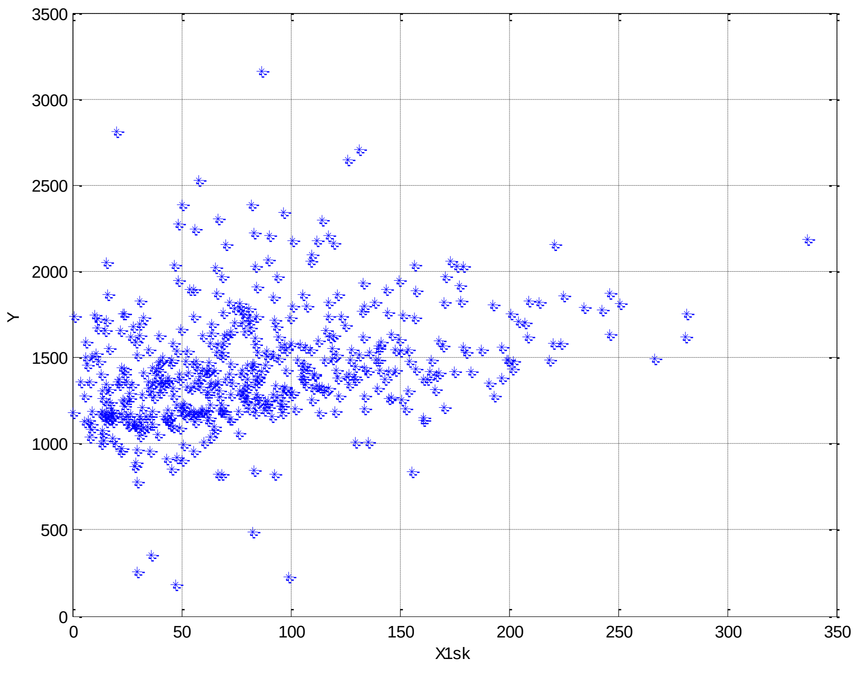

Figure 4.9: Y versus x_1sk of severe non-Normality Data set

4.5.2 Results on Artificial Datasets

We generate two artificial data sets according to the procedures introduced above. Each artificial data set has 500 projects. The detailed information regarding the two artificial datasets is presented in table 4.5. For a better assessment of accuracy, we apply an unequal split to the whole data set making the testing subset much larger than the other subsets.

Table 4.5: The partition of artificial data sets

Data Set	Sample size of Artificial Moderate non-Normality data	Sample size of Artificial Severe non-Normality data
Historical	50	50
Training	50	50
Testing	400	400
Total	500	500

We apply all the methods onto the two artificial data sets by following the same procedure used for real datasets. The results and comparisons are summarized as the following.

The results on artificial moderate non-Normality dataset are in table 4.6. It is shown that FWPSABE achieves the best performances in MMRE at 0.079 and MdMRE at 0.06 and the second best value 0.98 for PRED(0.25), while ANN gets the highest PRED(0.25) value at 0.99. Comparing the prediction error curves in fig 4.4 to the error curves in fig 4.10, it is observed that all the methods achieve much better performance on the artificial dataset and the differences among the candidate methods are much smaller on the artificial dataset. These findings imply that estimation methods in our study may converge to equally good prediction results on the moderately non-normal dataset with large size and FWPSABE is slightly better than other methods.

Table 4.6: The results and comparisons on artificial moderate non-Normality Dataset

Models	*MMRE*		*PRED(0.25)*		*MdMRE*	
	Training	Testing	Training	Testing	Training	Testing
ABE	0.068	0.116	0.98	0.94	0.048	0.093
FWABE	0.090	0.110	1.00	0.98	0.081	0.098
PSABE	0.057	0.086	1.00	0.98	0.043	0.068
FWPSABE	0.055	0.079	1.00	0.98	0.044	0.060
SVR	0.069	0.095	0.98	0.98	0.055	0.077
ANN	0.065	0.088	1.00	0.99	0.061	0.077
RBF	0.099	0.115	0.94	0.93	0.075	0.092
CART	0.099	0.109	0.98	0.95	0.074	0.090

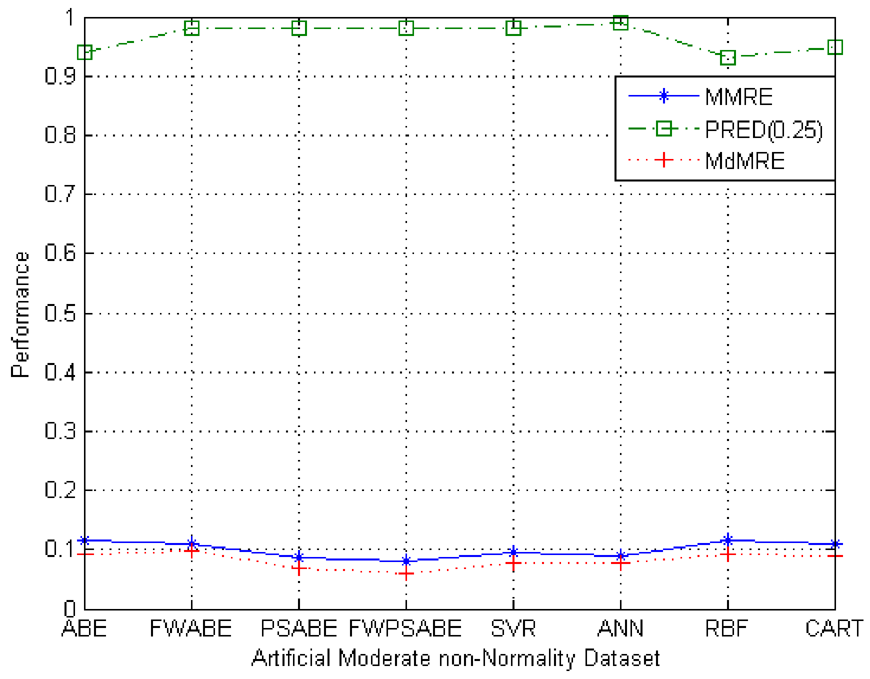

Figure 4.10: The testing results on Artificial Moderate non-Normality Dataset

Table 4.7 presents the results on artificial severe non-Normality dataset. FWPSABE achieves the best performances in MMRE at 0.15 and MdMRE at 0.10 and the second best value 0.80 for PRED(0.25), while CART obtains the highest *PRED*(0.25) value at 0.81. Comparing fig 4.11 to fig 4.10, it is shown that all the methods obtain poorer performances on severe non-normal dataset. This observation indicates that a high degree of non-normality has negative impacts on the performance of estimation methods in our study.

Table 4.7: The results and comparisons on Artificial Severe non-Normality Dataset

Models	*MMRE*		*PRED(0.25)*		*MdMRE*	
	Training	Testing	Training	Testing	Training	Testing
ABE	0.32	0.20	0.68	0.73	0.18	0.14
FWABE	0.34	0.19	0.72	0.77	0.14	0.13
PSABE	0.31	0.18	0.70	0.75	0.11	0.12
FWPSABE	0.30	0.15	0.74	0.80	0.14	0.10
SVR	0.34	0.18	0.62	0.76	0.19	0.12
ANN	0.34	0.17	0.70	0.79	0.16	0.12
RBF	0.37	0.18	0.66	0.80	0.18	0.13
CART	0.38	0.18	0.72	0.81	0.16	0.14

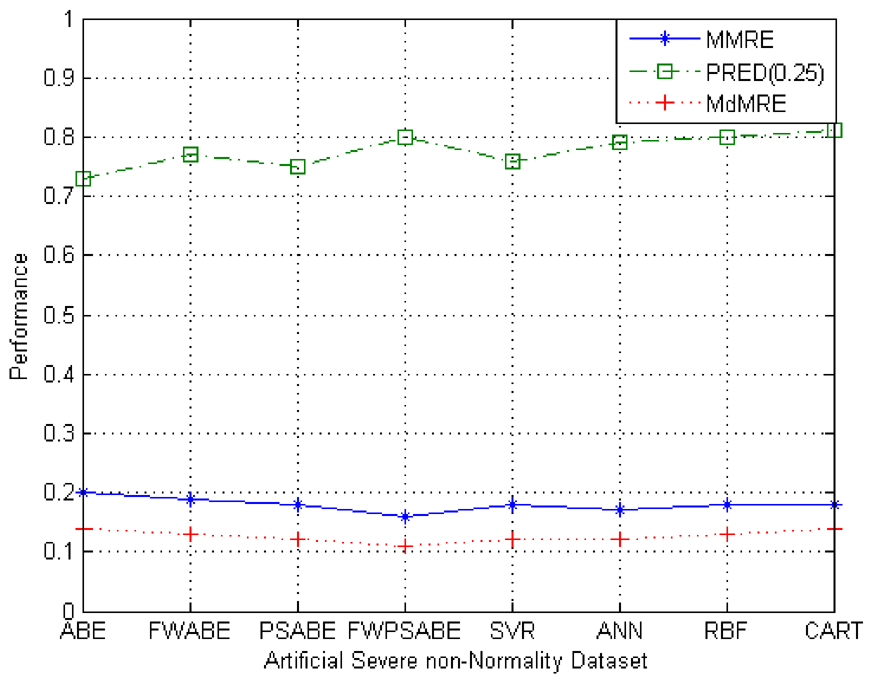

Figure 4.11: The testing results on Artificial Severe non-Normality Dataset

Chapter 5 Non-linear Adjustment by Artificial Neural Networks[3]

ABE predicts the cost of new project by retrieving similar historical projects. However, as mentioned in section 2.3.5, the retrieved solution has to be adjusted to adapt to the new situation. Several studies on the adjustment mechanisms are based on linear formula and restricted to numerical type of project features. On the other hand, software project datasets often exhibit non-normal characteristics together with large proportions of categorical features. To explore the possibilities for a better adjustment mechanism, this chapter proposes artificial neural network (ANN) for the non-linear adjustment of ABE (NABE) with the learning ability to adapt to complex relationships and to incorporate categorical features. The NABE is validated on four real world datasets and compared against the linear adjusted ABEs, CART, ANN and SWR. Moreover, eight artificial datasets are generated for a systematic investigation on the relationship between model accuracies and dataset properties. The comparisons and analysis show that non-linear adjustment could generally extend ABE's flexibility on complex datasets with large number of categorical features and improve the accuracies of ABE predictions.

[3] This chapter is related to the paper Li et al. 2009c

5.1 Introduction

The fundamental principle of ABE is simple: when provided a new project for estimation, the most similar historical projects (analogies) are retrieved, the solutions (cost values) of the retrieved projects are used to construct a 'retrieved solution' to the new project, with the expectation that the cost values of the retrieved projects will be similar to the real cost of the new project.

However, the adjustment on the retrieved solution is of necessity since it can capture the differences between the new project and the retrieved projects, and refine the retrieved solution into the target solution (Walkerden and Jeffery, 1999). In the literature, many types of adjustments have been proposed (refer to section 2.3.5). Most of these adjustments are based on predetermined linear forms without learning ability to adapt to more complex situations such as non-normality in the datasets. In addition, these adjustment techniques are limited to the numeric features despite that the categorical features also contain valuable information to improve the cost estimation accuracies (Angelis *et al.* 2000). In contrast, software project datasets often exhibit non-normal characteristics and contain large proportion of categorical features (Sentas and Angelis, 2006, Liu and Mintram, 2005).

To improve the existing adjustment mechanisms, we propose a more flexible non-linear adjustment with learning ability and including categorical features. The Non-linearity adjusted Analogy Based Estimation (NABE) is achieved by adding a non-linear component (Artificial Neural Network) onto the retrieved solution of the ABE system. In this approach, the ordinary ABE procedure is first executed to produce an un-adjusted retrieved solution to the new project. Then, the differences between the new project's features and its neighbors' features are used as inputs to ANN model to generate the non-linear adjustment. Finally, the retrieved solution and the adjustment from ANN are combined to form the final prediction.

The rest of this chapter is organized as follows: section 5.2 describes the details of the non-linearity adjusted ABE system (NABE). Section 5.3 introduces four real world data sets and the experiment design. In section 5.4, the NABE is tested on the real world datasets and is compared against the linear adjusted ABEs, ANN, CART and SWR. In section 5.5, eight artificial data sets are generated and a systematic analysis is conducted to explore how the model accuracies are related to dataset properties. The final section presents the discussions of this work.

5.2 Non-linearity Adjusted ABE System

5.2.1 Motivations

Analogy based software cost estimation is essentially a case-based reasoning (CBR) approach. Fig 5.1 illustrates the system diagram of ABE with adjustment form in the following formula:

$$\hat{C}_x = g(C_1, C_2, \ldots, C_K) \tag{5.1}$$

where $\hat{C}_x$ denotes the estimated cost for the new project *x*, C_i is the cost value of the *i*th closest analogy to project *x*, and *K* denotes the total number of nearest neighbors. The retrieved solution function (5.1) only includes the 'cost' values as its variables while other project features such as 'lines of source code' and 'function points' do not appear in this function. In the literature, several retrieved solution functions have been proposed, such as un-weighted mean, weighted mean, and median.

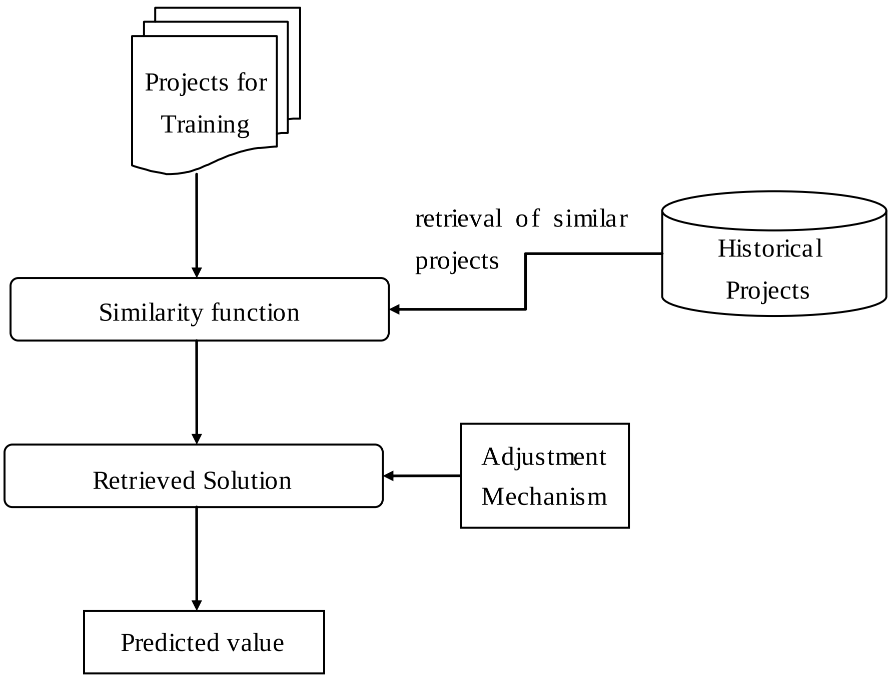

Figure 5.1: The general framework of analogy based estimation with adjustment

However, these solution functions can be rarely directly applied to predict $\hat{C}_x$. Instead, they need to be adjusted in order to fit the situations of the new project (Walkerden and Jeffery, 1999). Therefore the adjustment mechanisms should first identify the differences between the new project (features) and the retrieved projects (features) and then convert these differences into the amount of change in the cost value. In the literature, many adjustment techniques have been proposed (section 2.3.5).

Table 5.1: Comparison of published adjustment mechanisms

Source	Adjustment function	Adjustment feature	Categorical feature	Learning ability	Value of *K*
Walkerden and Jeffery (1999)	Linear	Function point (FP)	No	No	One
Mendes and Mosley (2003)	Linear	Size related features	No	No	Multiple
Jorgensen *et al.* (2003)	Linear	Function point (FP)	No	No	Multiple
Chiu and Huang (2007)	Linear	Size related features	No	Yes	One
Li *et al* (2007) and Li and Ruhe (2008)	Linear	All relevant features	Yes	No	Multiple

Table 5.1 characterizes each adjustment method from six aspects. The first column contains the source of the adjustment. The second column is the type of adjustment function (linear / non-linear). The third column describes the features used in the adjustment function. The fourth column indicates whether the categorical features are considered in the adjustment. The fifth column shows whether the adjustment function can learn from the training dataset to approximate a complex relationship. The last column presents the

number of nearest neighbors (one / multiple) used in the adjustment function. The reasons for selecting these criteria are as follows. The *function type* reflects the basic structure of the adjustment model. The *adjustment feature*, *categorical feature*, and *number of analogies* together determine the inputs of the adjustment model. The *learning ability* indicates whether the adjustment mechanism has the flexibility to adapt to complex relationships.

We can tell from Table 5.1 that most works are restricted to the linear functions without learning ability except the GA adjusted approach (Chiu and Huang, 2007). In addition, most adjustments do not consider the categorical features except the similarity adjusted function (Li *et al.* 2007, Li and Ruhe, 2007). To improve the adjustment mechanism, we propose a more flexible non-linear adjustment mechanism with learning ability and incorporating categorical features.

On the other hand, three relevant dataset characteristics are considered in our study: *non-normality*, *categorical feature*, and *dataset size*. These properties are likely to be relevant to the differences between the adjustment models. *Non-normality* is a frequently mentioned characteristic a cross the software engineering datasets (Pickard *et al.*, 2001). Many existing studies (Myrtveit and Stensrud, 1999, Shepperd and Kadoda, 2001, Mendes *et al.*, 2003) have considered the non-normality as an influential factor to the accuracies of the models including analogy based methods. Generally, a higher degree of non-normality leads to lower modeling accuracy. This property appears to be relevant to the *function type* of the adjustment models since the linear models usually work well under the normal condition and non-linear models with adaptive abilities seem to produce better results under non-normal conditions. Moreover, several applications of ANN in other research areas show that ANN model or ANN based models are robust to the non-normal datasets (Guh 2002, Chang and Ho 1999, Cannon 2007) and in theory ANN is capable of approximating arbitrary relationships (Lawrence, 1994). Therefore, it is expected that ANN based adjustment might enhance ABE model’s robustness to non-normality.

Given the fact that *categorical features* frequently appear in software engineering datasets (Sentas and Angelis, 2006, Liu and Mintram, 2005) and they may enclose useful information which could distinguish the projects (Angelis *et al.* 2000), many papers start to incorporate *categorical features* into consideration (Angelis *et al.* 2000, Sentas *et al.* 2005, Li *et al.* 2007, Li and Ruhe, 2008). However, most existing adjustment techniques do not consider categorical features. NABE aims to incorporate categorical features into the adjustment mechanism. Therefore, the appearance of categorical features is regarded as one important data set property in our study.

The *dataset size* is also an influential factor of ABE methods. The ABE system retrieves the similar cases from the historical dataset. The dataset with more projects may provide larger searching space for ABE. If the data is not very heterogeneous, it might lead to a higher chance for good prediction.

Several papers (Auer *et al.* 2006, Shepperd and Kadoda 2001, Shepperd and Schofield 1997) studied dataset size as one major factor on the accuracy of analogy based method. In both Shepperd and Schofield’s paper and Auer’s paper, the authors analyze the trends in estimation accuracy as the datasets grow, while Shepperd and Kadoda’s work confirms that ABE benefits from having larger training sets. In addition, Shepperd and Kadoda also find that ANN can achieve better results on larger training sets. Hence, the *dataset size* characteristic has some connections with the *learning ability* of both ANN and ABE.

As discussed above, dataset characteristics have large impacts on the estimation results and therefore it is more meaningful to identify which is the preferable estimation method in a particular context rather than to search for the ‘best’ prediction system for all cases. In the following sections, a detailed description of the non-linear adjusted analogy based estimation (NABE) is presented.

5.2.2 Artificial Neural Networks

First of all, the non-linear component of NABE - ANN is briefly introduced. Artificial neural network (ANN) is one type of machine learning technique that has played an important role in approximating complex relationships. Due to its excellent learning ability, ANN has been widely accepted for software cost estimation research.

In ANN architecture, there are typically three layers: the input layer, the hidden layers, and the output layer. All the layers are composed of neurons. The connections between neurons across layers represent the transmission of information between neurons. ANN has the following mathematical form:

$$y = y(\boldsymbol{x}) = \sum_{j=1}^{J} w_j f(\sum_{i=1}^{I} v_{ij} f(x_i) + \alpha_j) + \beta + \varepsilon \tag{5.2}$$

where $\boldsymbol{x}$ is an I-dimensional vector with $\{x_1, x_2, \ldots, x_I\}$ as its elements, $f(\bullet)$ is the user defined transfer function, ε is a random error with 0 as its mean value, J is the total number of hidden neurons, v_{ij} is the weight on the connection between the ith input neuron and the jth hidden neuron, α_j is the bias of the jth hidden neuron, w_j is the weight on the connection between the jth hidden neuron and the output neuron, and β is the bias of the output neuron. The weights and biases are determined by the training procedure minimizing the training error. The commonly used training error function Mean Square Error (MSE) is presented as follows:

$$E = \frac{1}{S}\sum_{s=1}^{S}(t^s - y^s)^2 \tag{5.3}$$

where y^s is the output of the network when the sth sample is the ANN input, and t^s is the sth training target. The classical Back Propagation (BP) algorithm is often used to update the weights and biases to minimize the training error.

As shown by formula (5.2), ANN has three user-defined parameters: the number of hidden layers, the number of hidden nodes and the type of transfer function. These parameters have major impacts on ANN's prediction performance (Martin *et al.* 1997). Among these parameters, one hidden layer is often recommended since multiple hidden layers may lead to an over parameterized ANN structure. For the number of hidden nodes, too few hidden nodes can hinder the network to approximate a desired function. On the contrary, too many hidden nodes can lead to over-fitting. The tuning of ANN parameters is given in section 5.3.2.

In our study, ANN is used as the adaptive non-linear adjustment component in NABE system. The NABE method and its system procedure are described in the following section.

5.2.3 Non-linear Adjusted Analogy Based System

From the explanations in section 5.2.1, the adjustment mechanism should capture the 'update' that transforms the solution from the retrieved projects into the target solution. Based on the linear adjustment model proposed by Chiu and Huang (2007), we extend the linear adjustment model to the following additive form:

$$\hat{C}_x = C_{w/o} + f(\boldsymbol{s}_x, \boldsymbol{S}_k) \tag{5.4}$$

where $f(\bullet)$ is an arbitrary function approximating the update that is necessary to change the retrieved solution into the target solution (in our study, $f(\bullet)$ is the ANN model), $\boldsymbol{s}_x$ is the feature vector of project x, $\boldsymbol{S}_k$ is the feature matrix of the K nearest neighbours and $C_{\text{w/o}}$ is the cost value obtained from the ABE without adjustment (or the retrieved solution).

The NABE system consists of two stages. In the first stage, the NABE system obtains the retrieved (un-adjusted) solution and trains the non-linear component – ANN. In the second stage the non-linear component is used to produce the update and then the update is added up to the retrieved solution to generate the final prediction.

Stage I - Training

The procedures of stage I are shown in fig 5.2. The jackknife approach (Angelis and Stamelos, 2000) (also known as leave one out cross-validation) is employed for the training of the non-linear adjustment (ANN). For each project in the training dataset, the following steps are performed:

Step 1: the *i*th project is extracted from the training dataset as the new project being estimated, and the remaining projects are treated as the historical projects in the ABE system.

Step 2: the ABE system finds the K nearest neighbors from the historical projects by the similarity measure. In this study, the Euclidean distance is used to construct the similarity function $Sim(i, j)$:

$$Sim(i,\ j) = 1 \Big/ \left[\delta + \sqrt{\sum_{q=1}^{Q} Dist(s_{iq}, s_{jq})} \right] \qquad \delta = 0.0001$$

$$Dist = \begin{cases} (s_{iq} - s_{jq})^2 & if\ s_{iq}\ and\ s_{jq}\ are\, numeric \\ 1, & if\ s_{iq}\ and\ s_{jq}\ are\, categorical\ and\ s_{iq} = s_{jq} \\ 0, & if\ s_{iq}\ and\ s_{jq}\ are\, categorical\ and\ s_{iq} \neq s_{jq} \end{cases} \tag{5.5}$$

where *i* represents the project being estimated, *j* denotes one historical project, s_{iq} is the *q*th feature value of project *i*, s_{jq} denotes the *q*th feature value of project *j*, *Q* is the total number of features in each project and $\delta = 0.0001$ is a small constant to prevent the situation that $\sqrt{\sum_{q=1}^{Q} Dist(s_{iq}, s_{jq})} = 0$. In our similarity function, we use un-weighted Euclidean distance to eliminate the impacts of different feature weights.

After obtaining the *K* nearest neighbors, the retrieved solution (cost value) to the *i*th project is generated. For the sake of simplicity, the un-weighted mean (Shepperd and Schofield, 1997) is used as the retrieved solution in this study.

Step 3: after obtaining the retrieved solution, the inputs and the training target are prepared to train the ANN model in (5.6). The inputs of ANN are the residuals between the features of project *i* and the features of its *K* nearest neighbors. The training target of ANN is the residual between the *i*th project's real cost value and the retrieved solution from its *K* nearest neighbors:

$$C_i - \sum_{k=1}^{K} \frac{C_k}{K} = \sum_{j=1}^{J} w_j\, f\Big(\sum_{k=1}^{K} \sum_{q=1}^{Q} v_{kqj}\, f(s_{iq} - s_{kq}) + \alpha_j\Big) + \beta + \varepsilon \tag{5.6}$$

The left hand side of (5.6) is the training target: the difference between the real cost of project i and the retrieved solution of project i. The right hand side of (5.6) is the ANN model with s_{iq} as the qth feature of project i, s_{kq} as the qth feature of its kth analogy (if s_{iq} and s_{kq} are categorical features, then s_{iq} - s_{kq} = 1 when $s_{iq} = s_{kq}$, and s_{iq} - s_{kq} = 0 when $s_{iq} \neq s_{kq}$), with w_j, v_{kqj}, α_j and β as ANN weights and biases, with $f(\bullet)$ as the transfer function, with J as the number of hidden neurons, with K as the total number of analogies, and with Q as the total number of features in each project. For example, if the ith project's real cost is 40 and the retrieved solution is 21, then the targeting output of ANN is 40 - 21 = 19.

Step 4: given the inputs and the targeting output, the Back Propagation (BP) algorithm is performed to update the parameters in (5.2) to minimize the training error MSE in (5.3).

After repeating the above procedure to all the projects in the training dataset, the training stage is completed and the system moves to the testing stage.

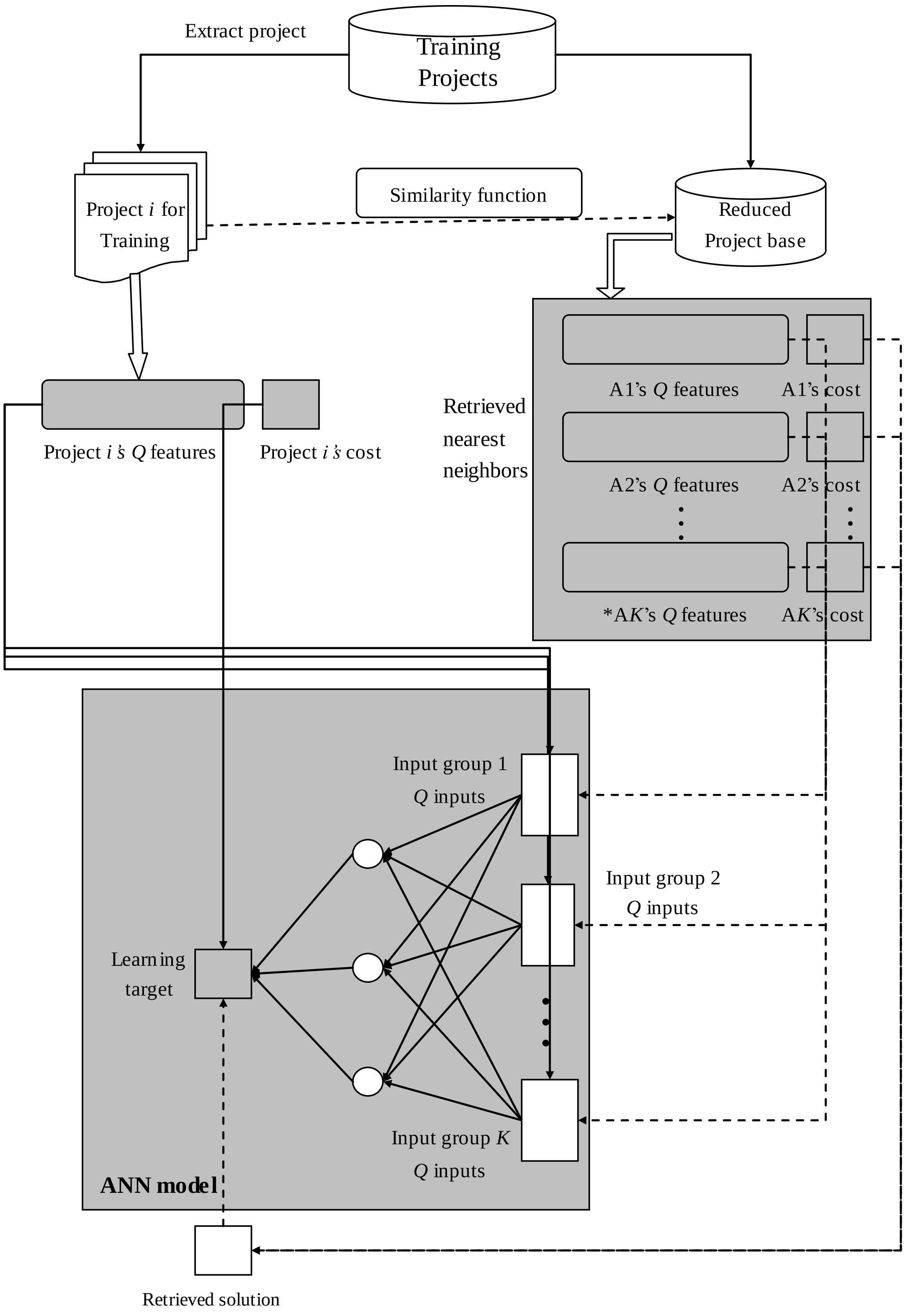

Figure 5.2: Training stage of the ANN adjusted ABE system with K nearest neighbors

* AK means the K^{th} nearest neighbor of project *i*.

Stage II - Predicting

The predicting stage is illustrated in fig 5.3. At this stage, a new project *x* is presented to the trained NABE system. Then, a set of *K* nearest neighbors are retrieved from the training dataset by applying (5.5) to calculate the similarities. After obtaining the *K* nearest neighbors, the retrieved solution function is used to generate the un-adjusted prediction, and the differences between features of project *x* and its *K* nearest neighbors are inputted into the trained ANN model to generate the adjustment. Finally, the ABE prediction and the ANN adjustment are summed up as the final prediction:

$$\hat{y}(x) = \sum_{k=1}^{K} \frac{C_k}{K} + \sum_{j=1}^{J} w_j f(\sum_{k=1}^{K} \sum_{q=1}^{Q} v_{kqj} f(s_{xq} - s_{kq}) + \alpha_j) + \beta \qquad (5.7)$$

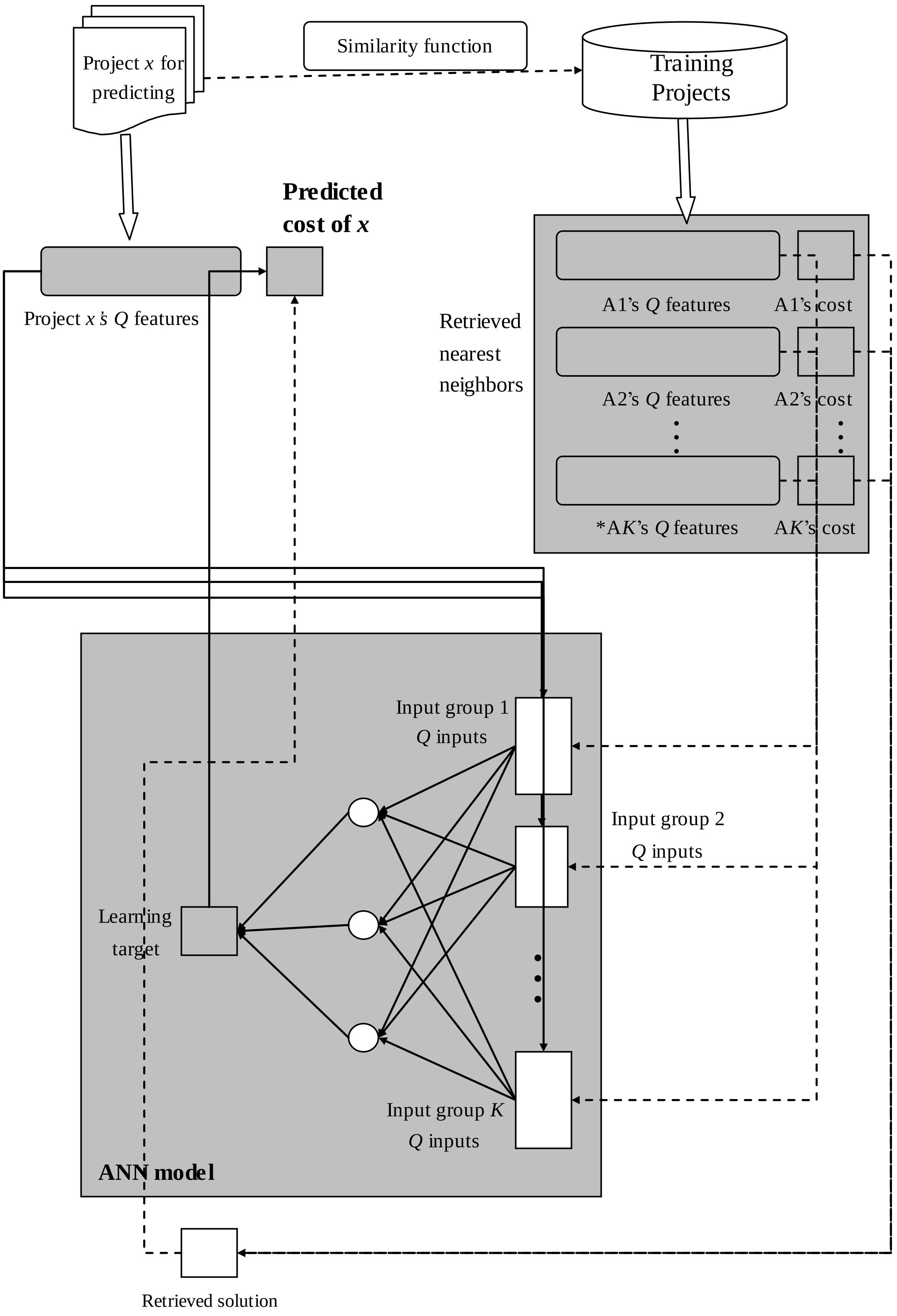

Figure 5.3: Predicting stage of the ANN adjusted ABE system with K nearest neighbors
* AK means the Kth nearest neighbor of project x.

5.3 Experiment Design

The data sets and experiments design are presented in this section.

5.3.1 Datasets

Four well known real world datasets are chosen for experiments. The Albrecht dataset is a popular dataset used by many recent studies (Shepperd and Schofield 1997, Heiat 2002, Auer *et al.* 2006). This dataset includes 24 projects developed by third generation languages. Eighteen out of 24 projects were written in COBOL, four were written in PL1, and two were written in DMS languages. There are five independent features: 'Inpcout', 'Outcount', 'Quecount', 'Filcount', and 'SLOC'. The two dependent features are 'Fp' and 'Effort'. The 'Effort' which is recorded in 1000 person hours is the targeting feature of cost estimation. The detailed descriptions of the features are shown in table B.1 in appendix. The descriptive statistics is presented in table B.2 in appendix. Among these statistics, the 'Skewness' and 'Kurtosis' are used to quantify the degree of non-normality of the features (Kendall and Stuart, 1976). It is noted that Albrecht is a relatively small dataset with high order non-normality compared to the other three datasets.

The Desharnais dataset was collected by Desharnais (1989). Despite the fact that the Desharnais dataset is relatively old, it is one of the large and publicly available datasets. Therefore it still has been employed by many recent research works, such as Mair *et al.* (2000), Burgess and Lefley, (2001), and Auer *et al.* (2006). This data set includes 81 projects (with 9 features) from one Canadian software company. Four of 81 projects contain missing values, so they have been excluded from further investigation. The 8 independent features are ‘TeamExp’, ‘ManagerExp’, ‘Length’, ‘Language’, ‘Transactions’, ‘Entities’, ‘Envergure’, and ‘PointsAdjust’. The dependent feature ‘Effort’ is recorded in 1000 *h*. The definitions of the features are provided in table B.3 in appendix. The descriptive statistics of all features are presented in table B.4 in appendix. It is shown that Desharnais is a larger dataset with relatively lower order non-normality compared with Albrecht dataset.

The Maxwell dataset (Maxwell, 2002) is a relatively new dataset and has already been used by some recent research works (Sentas *et al.*, 2005, Li et al. 2008b). This dataset contains 62 projects (with 26 features) from one of the biggest commercial banks in Finland. In this dataset, four out of 26 features are numerical and the remaining features are categorical. The categorical features can be further divided into ordinal features and nominal features, and they have to be distinguished. When calculating the similarity measure, the ordinal features are treated as ‘numerical features’ since they are sensitive to the order while the nominal features are regarded as ‘categorical’.

In Maxwell dataset, the numerical features are ‘Time’, ‘Duration’, ‘Size’ and ‘Effort’. The categorical features are ‘Nlan’, ‘T01’-‘T15’, ‘App’, ‘Har’, ‘Dba’, ‘Ifc’, ‘Source’ and ‘Telonuse’. The ordinal features are ‘Nlan’, and

'T01'-'T15'. The nominal features are 'App', 'Har', 'Dba', 'Ifc', 'Source' and 'Telonuse'. The definitions of all the features are presented in table B.5 in appendix. The descriptive statistics of all features are provided in table B.6 in appendix. It is shown that Maxwell is a relatively large dataset with relatively lower order non-normality and larger proportion of categorical features compared with Albrecht set and Desharnais set.

The ISBSG (International Software Benchmarking Standards Group) has developed and refined its data collection standard over a ten-year period based on the metrics that have proven to be very useful to improve software development processes. To the date of this study, the latest data release of this organization is the ISBSG R10 data repository (ISBSG 2007a) which contains a total of 4106 projects (with 105 features) coming from 22 countries and various organizations such as banking, communications, insurance, business services, government and manufacturing.

Due to the heterogeneous nature and the huge size of the entire repository, ISBSG recommends extracting out a suitable subset for any cost estimation practice (ISBSG, 2007b). At the first step, only the relevant features characterizing projects should be considered to create the subset. Thus, we select out 14 important features (including project effort) suggested by ISBSG (ISBSG 2007b): 'DevType', 'OrgType', 'BusType', 'AppType', 'DevPlat', 'PriProLan', 'DevTech', 'ProjectSize' (consisting of six sub features: 'InpCont', 'OutCont', 'EnqCont', 'FileCont', 'IntCont', and 'AFP'), and

'NorEffort'. The projects with missing values in any of the selected features are excluded from the subset. Then, a further step is taken to refine the subset. In ISBSG dataset, project data quality is rated and only projects with A or B rating are used in published research works. Therefore the projects with the ratings other than A and B are excluded from the subset. Moreover, since the normalized effort ('NorEffort') is used as the target for estimation, the risk of using normalized effort should be noted. For projects covering less than a full development life cycle, normalized effort is an estimate of the full development effort and this may introduce biasness. Hence the normalized ratio (normalized effort / summary effort) is used to refine the project subset. As suggested by ISBSG that a ratio of up to 1.2 is acceptable (ISBSG, 2007b), we filter out the projects with normalized ration larger than 1.2. Finally, the subset is further reduced to the projects with 'Banking' as 'OrgType'. All the above procedures results in a subset with 118 projects.

The definitions of the project features are presented in table B.7. The descriptive statistics of all features are summarized in table B.8. It is shown that the ISBSG subset is the largest dataset with high order non-normality and large proportion of categorical features compared with the datasets introduced above.

5.3.2 Experiment Design

Prior to the experiment setup, all types of features are normalized into [0, 1] by dividing each feature value by that feature range, similar to ANGEL (Shepperd and Schofield 1997). The three-fold cross-validation is used to assess the accuracies of the methods, similar to Jeffery *et al.* (2001), and Mendes *et al.* (2003).

Experiments procedures

After determining the cross-validation scheme, the following procedures are performed to validate the proposed NABE system with comparisons against other methods on each dataset.

1. The performances of NABE are analyzed on both training set and testing set by varying K values from 1 to 5 while keeping the similarity measure as the formula in (5.5) and the retrieved solution function as the 'un-weighted mean'. The reason for changing K values is that K is an important parameter which determines the number of inputs to the non-linear adjustment. The similarity measure and retrieved solution function are fixed because the focus of this study is on non-linear adjustment and these two parameters may not have direct impacts to the non-linear adjustment.
2. The optimal K value of the training practice (K minimizes the MMRE on training set) is selected to configure NABE for comparisons. Similarly,

the best variants of other methods on the training sets are also obtained. The training and testing results are summarized and analyzed.

3. The Wilcoxon signed-rank tests (α = 0.05) are performed to quantitatively identify the significance of difference of each pair-wised methods on testing sets.

Methods specifications

Many cost estimation techniques are included for comparisons. They are: the standard ABE (Shepperd and Schofield 1997), the Linear size adjusted ABE (LABE) (Walkerden and Jeffery 1999), Regression toward the mean adjusted ABE (RABE) (Jorgensen *et al.* 2003), GA optimized linear adjusted ABE (GABE) (Chiu and Huang 2007), Similarity adjusted ABE (SABE) (Li and Ruhe, 2007), and other popular cost estimation methods including the Classification and Regression Trees (CART) (Stensrud, 2001), the Artificial Neural Network (ANN) (Mair *et al.* 2000) and Stepwise Regression (SWR) (Mendes *et al.* 2003).

To eliminate the impacts from different parameters, all types of ABE methods are implemented with fixed similarity measure (Euclidean) and retrieved solution (un-weighted mean). The only changeable parameter *K* varies from 1 to 5. It is noted that, in SABE method the un-weighted similarity function is applied since the feature weighting is not included in this study.

For ANN, there are generally three parameters: the number of hidden nodes, the number of hidden layers and the types of hidden transfer functions. In our study, only one hidden layer is considered in order to avoid the over-parameterized ANN structure. The number of hidden nodes is chosen from the set {1, 3, 5, 7, 10} and the type of hidden transfer function is chosen from the set {Linear, Tan-Sigmoid, Log-Sigmoid}. Every combination of hidden node and hidden transfer function is evaluated on the training data. The optimal combination (minimizing MMRE) is used for testing and comparisons.

The CART (Brieman *et al.* 1984) is a non-parametric and tree structured analysis procedure that can be used for classification and regression. When the tree structure is applied for numerical targets they are often called regression trees. CART has the following advantages: the capability of dealing with categorical features, the easily understandable diagram of complex data and the ability to identify the major subsets in the total dataset (Srinivasan and Fisher 1995). The construction of the CART involves recursively splitting the data set into (usually two) relatively homogeneous subsets until the terminate conditions are satisfied. The best tree is obtained by applying cross-validation on the training set using a spread minimization criterion. The best tree model is used in testing and comparisons.

For the stepwise regression method (SWR), the optimal regression model is determined from the forward stepwise procedure on the training dataset. Then the optimal linear equation is used in testing and comparisons. When the categorical features appear in the dataset, the optimal scaling (or CATREG) technique by Angelis *et al.* (2001) is utilized to build the regression model based on both numerical and categorical features.

Finally, the random model (RAND) is also included in the comparisons as the control group to produce the estimation by randomly selecting any project's cost value from the dataset (training set or testing set).

All the methods are implemented via MATLAB code. The ANN component in the NABE system and the ANN method in comparisons are trained by BP algorithm. The mean squared error is used to determine how well the network is trained. The training stops when the MSE drops below the specified threshold = 0.01 in this study.

5.4 Results

In this section, the experimental results on four real world datasets are summarized and analyzed.

5.4.1 Results on Albrecht Dataset

Table 5.2 summarizes the three-fold cross validation results of NABE with different K values. It is observed that the setting $K = 4$ minimizes the training MMRE. Thus, the NABE system with $K = 4$ is chosen for the comparisons with other methods. In order to provide more insight on the magnitude of adjustment generated by ANN, the ratio of (absolute adjustment

/ non-adjusted cost value) is calculated across the testing sets. The mean value of these ratio values is 0.41 by the NABE system with *K* = 4.

Table 5.2: Results of NABE on Albrecht dataset

K value	Training			Testing		
	MMRE	PRED(0.25)	MdMRE	MMRE	PRED(0.25)	MdMRE
K = 1	0.84	0.13	0.64	0.70	0.50	0.28
K = 2	0.87	0.33	0.40	0.48	0.38	0.41
K = 3	0.89	0.46	0.28	0.59	0.46	0.29
K = 4	0.82	0.29	0.31	0.41	0.36	0.25
K = 5	0.93	0.42	0.29	1.01	0.33	0.39

Table 5.3 collects the training and testing results of the best variants of all cost estimation models. The configurations for ABE based methods are *K* = 2 for ABE, *K*= 3 for RABE, *K* = 1 for LABE, *K* = 2 for GABE and *K* = 1 for SABE. The testing results in table 5.3 show that the NABE achieves the best values in MMRE, PRED(0.25) and MdMRE. Among other types of ABEs, LABE obtains the smallest MMRE, ABE achieves the maximum PRED(0.25), and RABE has the minimal MdMRE. In addition, it is noted that all methods have better performances than the random model. Another interesting observation is that some testing results are better than the training results. Some published cost estimation works (such as Chiu and Huang (2007) and Huang and Chiu (2006)) also reported similar patterns. This may be due to the fact that the machine learning techniques are data driven methods and they learn from examples without any knowledge of the model type. If the testing

data happens to fit well to the model constructed on training data, then it is possible to have better testing results than training results.

Table 5.3: Accuracy comparison on Albrecht dataset

Methods	MMRE		PRED(0.25)		MdMRE	
	Training	Testing	Training	Testing	Training	Testing
NABE	0.82	0.41	0.29	0.36	0.31	0.25
RABE	0.85	0.66	0.37	0.21	0.36	0.45
LABE	0.81	0.61	0.29	0.21	0.39	0.53
GABE	0.92	0.77	0.40	0.33	0.45	0.48
SABE	0.84	0.81	0.33	0.25	0.41	0.46
ABE	0.93	0.87	0.29	0.33	0.46	0.43
ANN	0.97	0.85	0.46	0.33	0.30	0.39
CART	3.36	1.44	0.13	0.17	0.93	0.66
SWR	1.19	0.94	0.25	0.17	0.81	0.55
RAND	4.47	1.71	0.17	0.13	0.74	0.72

To further analyze the testing performances, we draw out the box plots of absolute residuals, because absolute residuals are less sensitive to bias than the asymmetric MRE values (Stensrud *et al.* 2003). The plots in fig 5.4 show that NABE has a lower median, a shorter inter-quartile range, and fewer outliers than other methods. It is also observed that the distributions of absolute residuals are heavily skewed. This implies that the standard *t*-test is no longer valid for significance testing. Thus, the assumption-free Wilcoxon signed-rank tests are performed instead.

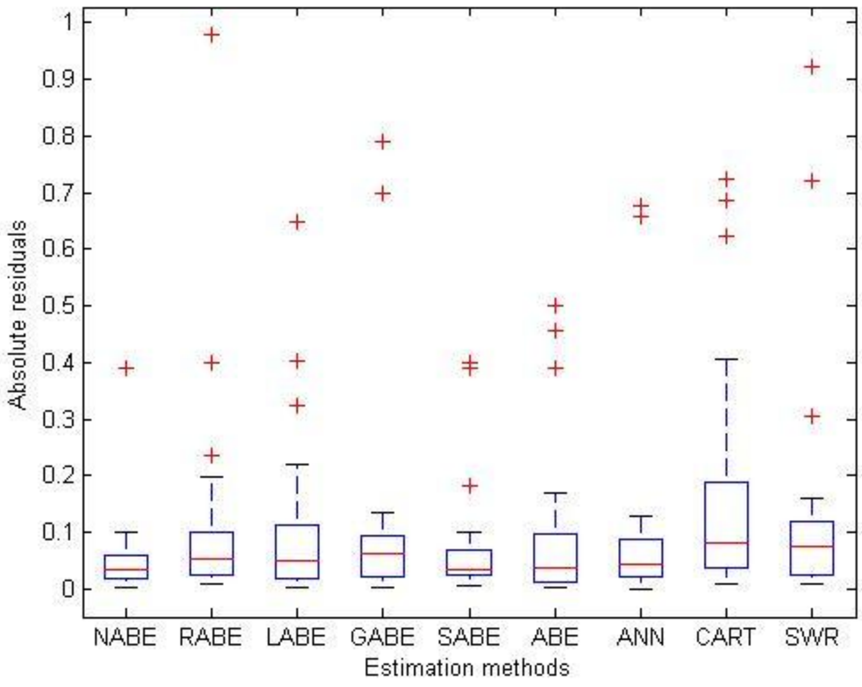

Figure 5.4: Boxplots of absolute residuals on Albrecht dataset

Table 5.4 summarizes the *p*-values of Wilcoxon tests of NABE versus other methods. Four paired comparisons have *p*-values smaller than 0.05. They are NABE v.s. RABE, NABE v.s. GABE, NABE v.s. CART, and NABE v.s. SWR. In addition, the improvements of NABE to other methods in terms of MMRE values are presented in Table 5.4. Four of the MMRE improvements are larger than 30% and the largest improvement is 60% on CART. The smallest improvement is 6% on LABE.

Table 5.4: NABE vs. other methods: *p*-values of the Wilcoxon tests and the improvements in percentages

	RABE	LABE	GABE	SABE	ABE	ANN	CART	SWR
p-value	0.02	0.19	0.04	0.08	0.08	0.12	0.00	0.00
Improvement on MMRE (%)	13	6	26	29	34	32	60	39

5.4.2 Results on Desharnais Dataset

In this section, we present the results on Desharnais dataset in a way that is similar to the analysis on Albrecht dataset. Table 5.5 illustrates the training errors and testing errors of NABE with respect to different *K* values. The setting *K* = 2 achieves the minimal training MMRE, and thus NABE with *K* = 2 is chosen for comparisons with other methods. The average of the ratios of (absolute adjustment / non-adjusted prediction) is 0.03 on the testing sets.

Table 5.5: Results of NABE on Desharnais dataset

K value	Training			Testing		
	MMRE	PRED(0.25)	MdMRE	MMRE	PRED(0.25)	MdMRE
K = 1	0.41	0.44	0.28	0.66	0.34	0.45
K = 2	0.39	0.40	0.35	0.52	0.36	0.32
K = 3	0.51	0.36	0.36	0.73	0.27	0.49
K = 4	0.52	0.30	0.42	0.64	0.21	0.50
K = 5	0.42	0.38	0.33	0.69	0.23	0.46

Table 5.6 summarizes the training and testing errors of the best variants of all cost estimation models. The optimal parameters for ABE based methods other than NABE are: ABE with *K* = 1, RABE with *K*= 1, LABE with *K* = 1, GABE with *K* = 2 and SABE with *K* = 4. The testing results show that NABE achieves smallest MMRE and MdMRE, and second largest PRED(0.25). Among other types of ABEs, GABE obtains the smallest MMRE, RABE achieves the largest PRED(0.25) and the minimal MdMRE. It is also observed that the differences between NABE and other methods are not as apparent as those on Albrecht dataset. This observation may be attributed to the

characteristic of Desharnais dataset: moderate non-normality. It implies that all methods tend to perform equally good when the data set is close to normal distribution. As to the control group, all other methods have better predictions than the random model.

Table 5.6: Accuracy comparisons on Desharnais dataset

Methods	MMRE		PRED(0.25)		MdMRE	
	Training	Testing	Training	Testing	Training	Testing
NABE	0.39	0.52	0.40	0.36	0.35	0.32
RABE	0.68	0.68	0.38	0.39	0.34	0.34
LABE	0.75	0.62	0.29	0.29	0.41	0.51
GABE	0.72	0.55	0.28	0.32	0.38	0.43
SABE	0.76	0.65	0.31	0.36	0.41	0.41
ABE	0.38	0.60	0.44	0.34	0.29	0.42
ANN	0.89	0.67	0.29	0.31	0.47	0.38
CART	0.58	0.71	0.31	0.25	0.41	0.44
SWR	0.67	0.73	0.35	0.35	0.39	0.34
RAND	1.81	1.14	0.12	0.18	0.67	0.60

For further analysis, the box plots of absolute residuals on testing datasets are presented in fig 5.5. The plots in fig 5.5 show that NABE's median is close to those of RABE, GABE, ANN and SWR, NABE has the shortest inter-quartile range, and NABE gets five outliers while SABE and CART have fewer ones though their outliers are more extreme. The distributions of absolute residuals are skewed and therefore Wilcoxon tests are used to quantitatively investigate the differences between NABE and other methods.

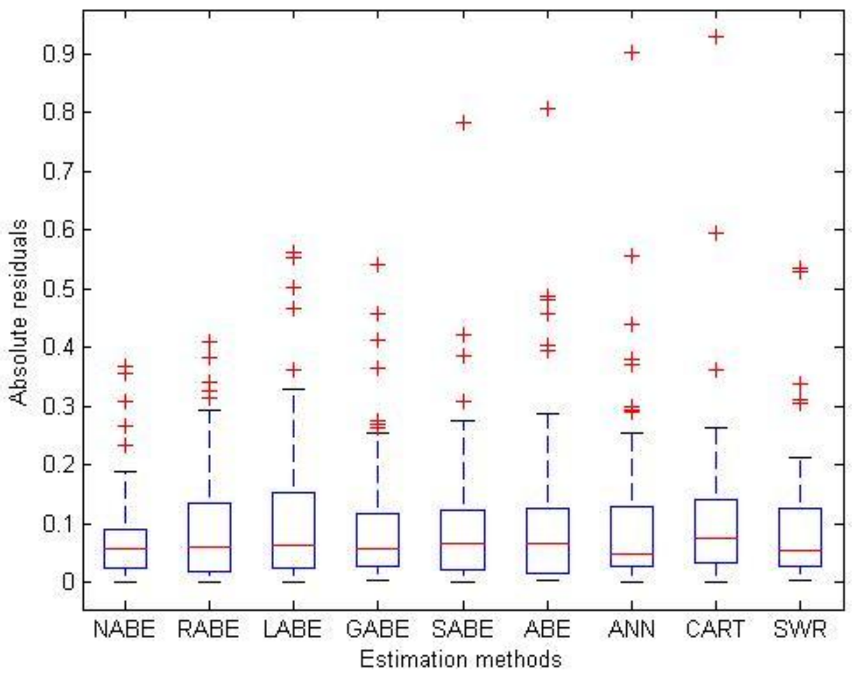

Figure 5.5: Boxplots of absolute residuals on Desharnais dataset

In table 5.7, the *p*-values from the Wilcoxon tests are presented together with the improvements on MMRE. Six out of eight *p*-values are larger than 0.05, and the remaining two *p*-values are NABE vs. LABE = 0.02 and NABE vs. CART = 0.03. All the MMRE improvements are not larger than 30%. The largest improvement is 30% on SWR while the smallest improvement is 7% on GABE. These observations confirm the previous observation that on Desharnais dataset, NABE does not perform significantly better than most methods.

Table 5.7: NABE vs. other methods: *p*-values of the Wilcoxon tests and the improvements in percentages

	RABE	LABE	GABE	SABE	ABE	ANN	CART	SWR
p-value	0.28	0.02	0.23	0.20	0.23	0.25	0.03	0.22
Improvement on MMRE (%)	24	17	7	20	14	23	27	30

5.4.3 Results on Maxwell Dataset

This section presents the results and comparisons on Maxwell dataset. Table 5.8 presents the three-fold cross validation results of NABE with different *K* values. The best setting *K* = 3 which minimizes the training MMRE is chosen for comparisons with other methods. The mean of the ratios of (absolute adjustment / non-adjusted prediction) is 0.37 on the testing sets.

Table 5.8: Results of NABE on Maxwell dataset

K value	Training			Testing		
	MMRE	PRED(0.25)	MdMRE	MMRE	PRED(0.25)	MdMRE
K = 1	0.91	0.23	0.61	1.21	0.16	0.57
K = 2	0.87	0.27	0.61	1.22	0.21	0.58
K = 3	0.80	0.23	0.51	0.80	0.35	0.45
K = 4	0.89	0.21	0.54	0.77	0.19	0.49
K = 5	0.89	0.24	0.56	0.93	0.19	0.56

Table 5.9 presents the training and testing accuracies of different cost estimation models. The results from best variants of all methods are collected in this table. The configurations for ABE based methods are: ABE with *K* = 3, RABE with *K* = 3, LABE with *K* = 2, GABE with *K* = 3 and SABE with *K* = 4. The results show that NABE achieves the best testing MMRE, PRED(0.25) and MdMRE. Among other types of ABEs, SABE obtains the smallest MMRE, LABE achieves the largest PRED(0.25), and SABE has the minimal MdMRE. As to the control group, all other methods seem to be better than the random model.

Table 5.9: Accuracy comparisons on Maxwell dataset

Methods	MMRE		PRED(0.25)		MdMRE	
	Training	Testing	Training	Testing	Training	Testing
NABE	0.80	0.80	0.23	0.35	0.51	0.45
RABE	0.78	0.88	0.23	0.16	0.57	0.62
LABE	0.74	1.08	0.24	0.31	0.44	0.63
GABE	0.92	0.98	0.21	0.26	0.45	0.52
SABE	0.94	0.85	0.15	0.23	0.60	0.50
ABE	0.92	1.04	0.23	0.21	0.63	0.62
ANN	1.19	1.32	0.34	0.13	0.52	0.62
CART	1.60	1.52	0.23	0.26	0.61	0.65
SWR	1.53	1.09	0.18	0.23	0.65	0.76
RAND	2.49	1.70	0.16	0.05	0.66	0.81

To further analyze the testing results, we draw out the box plots of absolute residuals. The plots in Fig 5.6 show that NABE has a median close to those of GABE and SABE; NABE has an inter-quartile range close to those of GABE, SABE and CART; NABE gets five outliers while RABE, GABE, ABE, ANN and SWR have fewer outliers though some of their outliers are more extreme. The distributions of absolute residuals suggest using the Wilcoxon tests to identify the differences between NABE and other methods.

Table 5.10 summarizes the *p*-values of Wilcoxon tests and the improvements on MMRE values. Four out of eight *p*-values are smaller than 0.05. Two of the MMRE improvements are larger than 30%. The largest improvement is 48% on CART and the smallest improvement is 7% on GABE. These observations confirm the finding that NABE performs significantly better than other methods except SABE and GABE, on Maxwell dataset.

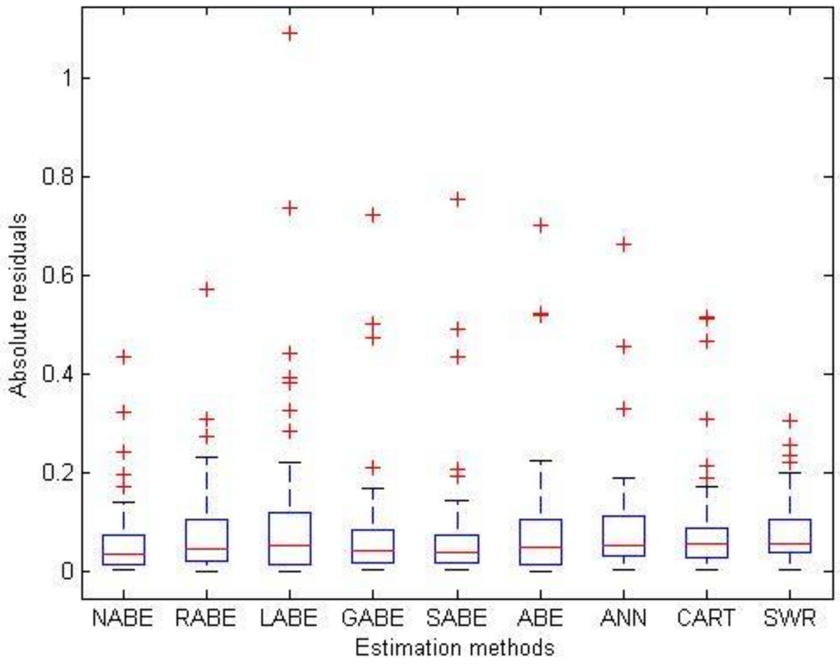

Figure 5.6: Boxplots of absolute residuals on Maxwell dataset

Table 5.10: NABE vs. other methods: *p*-values of the Wilcoxon tests and the improvements in percentages

	RABE	LABE	GABE	SABE	ABE	ANN	CART	SWR
p-value	0.09	0.02	0.14	0.15	0.06	0.00	0.02	0.02
Improvement on MMRE (%)	11.00	27.00	7.00	20.00	24.00	40.00	48.00	28.00

5.4.4 Results on ISBSG Dataset

In this section, we present the results and comparisons on ISBSG dataset. Table 5.11 illustrates the training and testing errors of NABE with different K values. The setting $K = 2$ achieves the minimal training MMRE and therefore NABE with $K = 2$ is chosen for comparisons with other methods. The mean value of the ratios of (absolute adjustment / non-adjusted prediction) is 0.43 on the testing sets, which is close to that of Albrecht data set and Maxwell data set. Table 5.12 summarizes the comparisons among the best variants of different cost estimation models. The optimal parameters for ABE based methods are: ABE with K = 3, RABE with K = 3, LABE with K = 1, GABE with K = 3 and SABE with K = 5. The results show that the NABE achieves the best testing MMRE, PRED(0.25), and MdMRE. Among other types of ABEs, SABE obtains the smallest MMRE, RABE achieves the largest PRED(0.25) and the minimal MdMRE. As to the control group, all methods appear to be better than the random model.

Table 5.11: Results of NABE on ISBSG dataset

K value	Training			Testing		
	MMRE	PRED(0.25)	MdMRE	MMRE	PRED(0.25)	MdMRE
K = 1	0.98	0.25	0.60	0.87	0.26	0.55
K = 2	0.89	0.33	0.46	0.74	0.31	0.42
K = 3	0.97	0.26	0.53	0.89	0.22	0.49
K = 4	1.00	0.15	0.63	0.95	0.22	0.58
K = 5	1.10	0.10	0.69	1.03	0.23	0.61

Table 5.12: Accuracy comparisons on ISBSG dataset

Methods	MMRE		PRED(0.25)		MdMRE	
	Training	Testing	Training	Testing	Training	Testing
NABE	0.89	0.74	0.33	0.30	0.46	0.42
RABE	1.16	1.36	0.28	0.28	0.51	0.54
LABE	1.19	1.13	0.29	0.17	0.53	0.58
GABE	1.13	1.09	0.25	0.21	0.54	0.60
SABE	0.91	0.85	0.23	0.18	0.51	0.58
ABE	0.97	0.98	0.16	0.22	0.63	0.59
ANN	0.82	0.96	0.27	0.25	0.49	0.60
CART	1.26	1.07	0.19	0.18	0.73	0.61
SWR	0.77	0.82	0.29	0.19	0.54	0.60
RAND	2.17	2.29	0.13	0.09	0.73	0.70

The box plots of absolute residuals on testing sets are provided for further analysis. The plots in fig 5.7 show that NABE achieves a lower median, the shorter inter-quartile range than other methods. Another observation is that all methods are prone to extreme outliers. This may be attributed to the fact that ISBSG dataset was collected inter-organizationally and internationally. Due to the diverse sources of data, even two similar projects might have significantly different amounts of cost. In the next step, Wilcoxon tests are used to assess the differences between NABE and other methods.

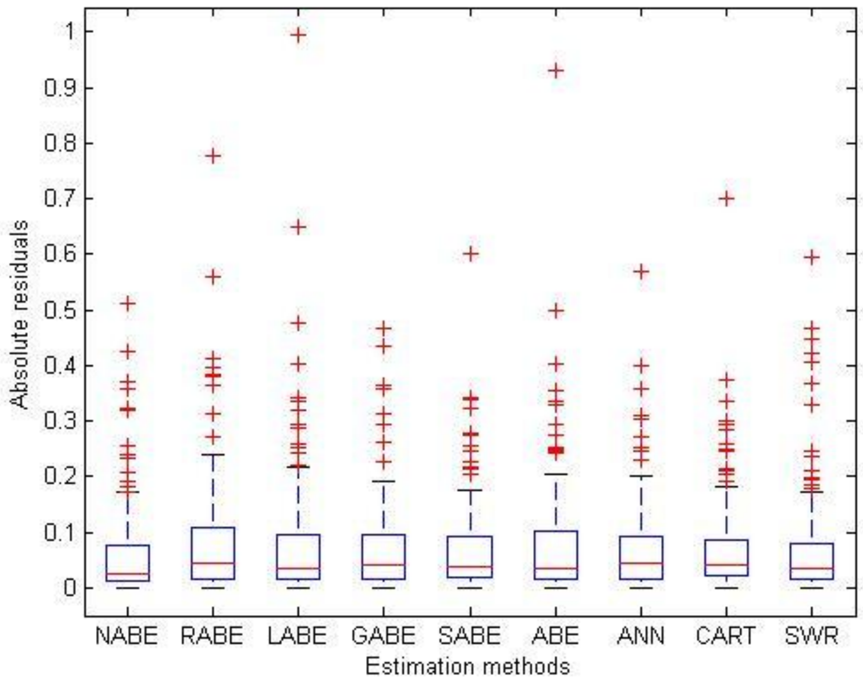

Figure 5.7: Boxplots of absolute residuals on ISBSG dataset

In table 5.13, the *p*-values from the Wilcoxon tests are presented together with the improvements on MMRE. In this table, all *p*-values are not larger than 0.05. As for the MMRE improvement, four MMRE improvements are larger than 30%. The largest improvement is 48% on RABE while the smallest improvement is 14% on SWR.

Table 5.13: NABE vs. other methods: *p*-values of the Wilcoxon tests and the improvements in percentages

	RABE	LABE	GABE	SABE	ABE	ANN	CART	SWR
p-value	0.02	0.05	0.04	0.05	0.02	0.02	0.01	0.02
Improvement on MMRE (%)	48	31	35	17	29	27	34	14

5.5 Analysis on Dataset Characteristics

In section 5.4, results and comparisons are presented on each real dataset individually. However, the results vary significantly from one dataset to another. For instance, NABE is statistically better than RABE on ISBSG dataset (p = 0.02) but their performances are similar to each other statistically on Desharnais dataset (p = 0.28). This is probably due to the fact that model accuracies are not only affected by the parameters selections but also affected by other factors such as the dataset characteristics (Shepperd and Kadoda, 2001). In this section, we conduct a systematic investigation in order to explore the relationship between model accuracy and the dataset characteristics, and identify under which conditions NABE is the preferred prediction system and under what conditions other methods can also be recommended.

Table 5.14 summarizes a set of characteristics of the real world datasets. The columns in this table list the dataset name, the number of projects, the total number of features, the number of categorical features, and the average value of absolute skewness and the average of kurtosis of each feature. The skewness and kurtosis values together reflect the degree of non-normality of the dataset.

Table 5.14: Characteristics of the four real world datasets

Dataset	Number of Projects	Number of Features	Number of Categorical Features	Avg. Skewness	Avg. Kurtosis
Albrecht	24	7	0	2.03	7.27
Desharnais	77	9	1	1.18	5.03
Maxwell	62	26	6	0.97	5.42
ISBSG	118	14	7	1.67	7.42

This table provides some insights to each dataset. It is shown that software datasets often exhibits a mixture of several characteristics such as skewness and excessive outliers (kurtosis). These characteristics do not always appear in the same degree. In some cases they are moderate such as the Albrecht dataset, while in other cases they are severe such as the ISBSG dataset. It is also noted that the data sets are largely contrasting to each other, for example Albrecht dataset has a relatively small size and small proportion of categorical features while Maxwell dataset is larger and has a large proportion of categorical features. However, based on only the real world datasets, there are still some difficulties for a systematic analysis. The real dataset properties are uncontrollable and the real world datasets cannot cover the full range of the combinations of the properties being studied.

Artificially generated dataset by simulation (Pickard *et al.* 1999, Shepperd and Kadoda 2001) is a feasible solution to the above difficulties. This approach generates artificial dataset from predefined distributions and equations. The simulated dataset provides the researcher with more control over the characteristics of a dataset. It especially enables the researcher to vary one property at a time and thus allows a more systematic exploration of the relationship between dataset characteristics and model accuracies. As a simple but powerful tool for empirical evaluations, this technique has been frequently implemented by several recently published studies (Myrtveit *et al.* 2005, Li *et al.* 2008a).

Besides the simulation approach, bootstrapping (Efron and Gong 1983) is often used to produce artificial datasets to study the uncertainties in the predictions (Angelis and Stamelos, 2000). Its principle is to generate several new datasets with the same size as the original dataset by randomly sampling original data with replacement. Each new dataset may have some items from the original dataset appearing more than once while some not appearing at all. However, bootstrapping is not considered for artificial dataset generation in this study. The reason is that our study mainly emphasizes on varying dataset properties to investigate the relationships between dataset properties and model accuracies but bootstrapping only generates a series of datasets based on original data and offers limited variability for changing the dataset properties. Also, the simulation technique provides a more explicit control over the dataset properties such as adjusting the distribution parameters to vary the skewness and kurtosis of the variable distribution.

In section 5.5.1, we simulate 8 artificial datasets to match the 8 different combinations of the 3 data characteristics. Due to the computational limits, we only considered two levels for each characteristic: such as Large/Small for the 'Dataset size', Large/Small for the 'Proportion of categorical features', and Severe/Moderate for the 'Non-normality'.

5.5.1 Artificial Dataset Generation

In this section, we present the procedures of artificial datasets generation. We extend Pickard's equation of artificial dataset generation in this work. Other types of simulation techniques for artificial dataset generation are also available in the literature. For more details, readers can refer to Shepperd and Kadoda (2001), Foss *at al.* (2003), and Myrtveit *et al.* (2005).

Based on Pickard's method, we simulate the combinations of characteristics from the equation (5.8):

$$y = 1000 + 6x_1sk + 3x_2sk + 2x_3sk + 5x_4sk + 10x_5sk + x_6sk + e \qquad (5.8)$$

The independent variables are x_1sk, x_2sk, x_3sk, x_4sk, x_5sk, and x_6sk. Among them, x_1sk, x_2sk, and x_3sk are continuous variables, and x_6sk is a categorical variable. The first variable x_1sk is treated as the feature 'function point' for the linear adjustment methods. The last term e in (5.8) is the normally distributed noise with mean 0 and variance 1. To simulate different proportions of categorical features (Large/Small), x_4sk and x_5sk are defined as categorical variables for the situation of large proportion (50%) while x_4sk and x_5sk are set to be continuous to represent the situation of small proportion of categorical features (16.7%).

The non-normality is represented by skewness and outliers (kurtosis). For the continuous variables, the skewnesses are generated by five independent Gamma distributed random variables x_1', x_2', x_3', x_4', and x_5'with scale parameter $\theta = 2$ and shape parameter $k = 3$ representing moderate skewness, and $\theta = 2$ and $k = 1$ for the severe skewness. For the categorical variables, the moderate skewnesses are simulated by the independent discrete random variables x_4', x_5', and x_6' with the distribution $\{P(X = 1) = 0.1; P(X = 2) = 0.1, P(X = 3) = 0.5, P(X = 4) = 0.2, P(X = 5) = 0.1\}$ and the severe skewnesses are simulated by the distribution $\{P(X = 1) = 0.7; P(X = 2) = 0.1, P(X = 3) = 0.1, P(X = 4) = 0, P(X = 5) = 0.1\}$. To vary the scale of the independent variable, we then multiply x_1' by 10 to create variable x_1sk, x_2' by 3 to create the variable x_2sk, x_3' by 20 to create the variable x_3sk, x_4' by 5 to create the variable x_4sk, x_5' by 2 to create the variable x_5sk, and x_6' by 1 to create the variable x_6sk. The outliers are generated by multiplying or dividing the dependent variable *y* by a constant. We select 1% of the data points to be the outliers. Half of the outliers are obtained by multiplying while half of them are obtained by dividing. For the moderate outliers, we set the constant value as 2, while for the severe outliers, 6 is chosen to be the constant.

For dataset sizes, we generate 400 projects to form the large sized dataset and 40 projects to construct the small sized dataset. Table 5.15 summarizes the properties of the 8 artificial datasets.

Table 5.15: Artificial datasets and properties

Dataset ID	Size (number of projects)	Number of Categorical features (proportion)	Degree of Non-normality (Avg. skewness, Avg kurtosis)
#1	Small (40)	Small (16.7%)	Moderate (0.75, 3.10)
#2	Small (40)	Small (16.7%)	Severe (2.32, 9.87)
#3	Small (40)	Large (50%)	Moderate (0.61, 3.37)
#4	Small (40)	Large (50%)	Severe (2.84, 9.71)
#5	Large (400)	Small (16.7%)	Moderate (0.93, 3.72)
#6	Large (400)	Small (16.7%)	Severe (3.21, 13.9)
#7	Large (400)	Large (50%)	Moderate (0.82, 3.63)
#8	Large (400)	Large (50%)	Severe (3.32, 10.09)

5.5.2 Comparisons on Modeling Accuracies

The experimental procedures presented in section 5.3 are applied on all artificial datasets. The comparisons between NABE and other models are presented first, since the relative performances of NABE to other methods could provide more insights about how to choose an appropriate cost estimation method under a certain condition. Table 5.16 summarizes the results of Wilcoxon signed rank tests. These significance tests assess the differences between the absolute residuals of NABE's predictions and the absolute residuals of other methods' predictions. The confidence limit is set at α = 0.05. In Table 5.16, the entry with 'Y' indicates that NABE performs significantly better than the method located in this entry's corresponding column. The last column summarizes the total number of 'Y's in each row (dataset).

Table 5.16: Comparative performance of NABE to other methods

Dataset ID	RABE	LABE	GABE	SABE	ABE	ANN	CART	SWR	Totals
#1									0
#2		Y						Y	2
#3	Y	Y				Y			3
#4	Y	Y	Y	Y	Y	Y	Y	Y	8
#5		Y	Y						2
#6		Y						Y	2
#7	Y	Y	Y		Y				4
#8	Y	Y	Y	Y	Y	Y	Y	Y	8

The results in table 5.16 show that NABE achieves better performance than all other methods on datasets #4 and #8. Both have large proportions of categorical features and severe non-normality. This observation suggests that NABE might be the best choice among all methods in our study, when the dataset is highly non-normal and with large proportion of categorical features. This observation also confirms the findings on ISBSG dataset which has similar properties to dataset #8. Another interesting observation is that NABE obtains the equally good predictions as other methods on dataset #1 which has small size, small number of categorical features and moderate non-normality.

When Compared to the real world datasets, Dataset #1's properties are closest to those of Desharnais set on which NABE also performs equally as other methods except LABE and CART.

The analysis above clarifies the conditions under which NABE is preferable to other methods. To further study the relationship between dataset property and model accuracy, we analyze the model predictions under single dataset characteristic.

5.5.3 Analysis on 'Size'

Table 5.17 summarizes the testing MMREs of each cost estimation model on the artificial datasets grouped under different 'size'. The results show that NABE achieves the lowest MMREs on datasets #2, #4, #5, #6, #7, and #8. It is also observed that the dataset size might largely influence the prediction accuracies. More specifically, almost all the methods obtain smaller MMRE values on larger datasets.

Table 5.17: Testing MMREs under different dataset size

Dataset	Size	NABE	RABE	LABE	GABE	SABE	ABE	ANN	CART	SWR
#1	Small	0.13	0.13	0.13	0.12	0.10	0.15	0.10	0.13	0.17
#2		0.14	0.19	0.25	0.14	0.15	0.18	0.14	0.22	0.44
#3		0.19	0.28	0.27	0.24	0.18	0.19	0.28	0.24	0.15
#4		0.28	0.45	0.44	0.48	0.45	0.47	0.41	0.44	0.68
#5	Large	0.08	0.15	0.17	0.18	0.08	0.09	0.08	0.09	0.14
#6		0.10	0.14	0.19	0.13	0.14	0.12	0.12	0.17	0.47
#7		0.08	0.23	0.20	0.21	0.13	0.20	0.14	0.12	0.12
#8		0.24	0.40	0.32	0.35	0.35	0.39	0.34	0.45	0.65

To further investigate the 'size' property, we compare the absolute residuals of predictions using the small datasets and the large datasets. The difference is tested by using the Mann-Whitney U test setting the confidence limit at $\alpha = 0.05$, since the sample sizes are not equal (40 data points vs. 400 data points). The results are presented in Table 5.18. The entry with 'Y' means the difference between the datasets pair in its row is significant when using the model in its column. Table 5.18 shows that a larger dataset size may significantly reduce prediction error measured by absolute residuals. Most approaches including NABE could benefit from having larger datasets. However, SWR seems to be not influenced by the dataset size. This may be attributed to the fact that SWR constructs the regression line from the data with only a few critical data points. This finding also confirms the suggestion from Shepperd and Kadoda (2001) that for the machine learning methods, large dataset size could reduce the prediction errors when other properties are fixed.

Table 5.18: Mann-Whitney U tests of dataset size influences

Datasets pair	NABE	RABE	LABE	GABE	SABE	ABE	ANN	CART	SWR
#1 vs. #5	Y			Y		Y			
#2 vs. #6	Y	Y	Y			Y		Y	
#3 vs. #7	Y	Y	Y		Y		Y	Y	
#4 vs. #8	Y	Y	Y	Y	Y	Y	Y		

5.5.4 Analysis on 'Proportion of categorical features'

This section presents the analysis on the proportion of categorical features. Table 5.19 is essentially a re-arrangement of the rows in table 5.17. In table 5.19, the artificial datasets are grouped under different 'proportion of categorical features'. It is observed that large proportion of categorical features may have negative impacts on the prediction accuracy. This finding is reflective of the fact that categorical features may have less statistical power compared with numerical features (Kirsopp *et al.* 2003).

Table 5.19: Testing MMREs under different proportions of categorical features

Dataset	Proportion of categorical features	NABE	RABE	LABE	GABE	SABE	ABE	ANN	CART	SWR
#1	Small	0.13	0.13	0.13	0.12	0.10	0.15	0.10	0.13	0.17
#2		0.14	0.19	0.25	0.14	0.15	0.18	0.14	0.22	0.44
#5		0.08	0.15	0.17	0.18	0.08	0.09	0.08	0.09	0.14
#6		0.10	0.14	0.19	0.13	0.14	0.12	0.12	0.17	0.47
#3	Large	0.19	0.28	0.27	0.24	0.18	0.19	0.28	0.24	0.15
#4		0.28	0.45	0.44	0.48	0.45	0.47	0.41	0.44	0.68
#7		0.08	0.23	0.20	0.21	0.13	0.20	0.14	0.12	0.12
#8		0.24	0.40	0.32	0.35	0.35	0.39	0.34	0.45	0.65

Table 5.20 presents the results of Wilcoxon signed rank tests with confidence level at $\alpha = 0.05$ on the absolute residuals of predictions using the datasets with smaller number of categorical features and the datasets with larger number of categorical features.

Table 5.20: Wilcoxon tests of proportion of categorical features influences

Datasets pair	NABE	RABE	LABE	GABE	SABE	ABE	ANN	CART	SWR
#1 vs. #3		Y	Y	Y			Y	Y	
#2 vs. #4	Y	Y	Y	Y	Y	Y	Y	Y	Y
#5 vs. #7						Y			
#6 vs. #8	Y	Y	Y	Y	Y	Y	Y	Y	Y

In general, all methods are more or less affected by this property. Among them, NABE, SABE and SWR are least sensitive to the categorical values. The probable reason is that CATREG technique is adopted in SWR model, and NABE and SABE both can make use of the categorical features in their adjustment mechanism.

5.5.5 Analysis on 'Degree of non-normality'

This section provides the analysis on degree of non-normality. Table 5.21 is also a re-arrangement of the rows in table 5.17. In table 5.21, the artificial datasets are grouped under different 'degree of non-normality'. It is noted that most methods obtain larger MMRE values under severe non-normal conditions. This indicates a trend that the increase of non-normality may result in a decrease of the prediction accuracy. However, NABE appeared to be least sensitive to non-normality while SWR seems to be most sensitive to non-normality. This observation supports our argument in section 5.2 that ANN could enhance ABE's robustness to non-normal data.

Table 5.21: Testing MMREs under different degrees of non-normality

Dataset	Non-normality	NABE	RABE	LABE	GABE	SABE	ABE	ANN	CART	SWR
#1	Moderate	0.13	0.13	0.13	0.12	0.10	0.15	0.10	0.13	0.17
#3		0.19	0.28	0.27	0.24	0.18	0.19	0.28	0.24	0.15
#5		0.08	0.15	0.17	0.18	0.08	0.09	0.08	0.09	0.14
#7		0.08	0.23	0.20	0.21	0.13	0.20	0.14	0.12	0.12
#2	Severe	0.14	0.19	0.25	0.14	0.15	0.18	0.14	0.22	0.44
#4		0.28	0.45	0.44	0.48	0.45	0.47	0.41	0.44	0.68
#6		0.10	0.14	0.19	0.13	0.14	0.12	0.12	0.17	0.47
#8		0.24	0.40	0.32	0.35	0.35	0.39	0.34	0.45	0.65

Table 5.22 presents the results of Wilcoxon signed rank tests with confidence level at $\alpha = 0.05$ on the absolute residuals of predictions using moderate non-normal datasets and severe non-normal datasets. The results confirm the finding that NABE is least sensitive to non-normality while SWR is most sensitive to the non-normal property. Table 5.22 also can partially support Shepperd and Kadoda's (2001) argument that ABE is preferred to SWR if the dataset contains large proportion of outliers.

Table 5.22: Wilcoxon tests of non-normality influences

Datasets pair	NABE	RABE	LABE	GABE	SABE	ABE	ANN	CART	SWR
#1 vs. #2			Y						Y
#3 vs. #4		Y	Y	Y	Y	Y	Y	Y	Y
#5 vs. #6									Y
#7 vs. #8	Y	Y		Y	Y	Y	Y	Y	Y

5.6 Discussions

To improve the adjustment mechanism, this chapter presents a flexible non-linear adjustment mechanism with learning ability and incorporating categorical features. The non-linearity adjusted Analogy Based Estimation (NABE) is implemented by adding a non-linear component (Artificial Neural Network) onto the retrieved solution of the ABE system. The proposed NABE is validated on four real world datasets with the comparisons against the published linear adjusted ABEs and three well established methods: CART, ANN and SWR. The results and comparisons show that NABE generally achieves best MMRE, PRED(0.25) and MdMRE values on the real world datasets.

To answer the question: under what conditions NABE is preferred, we generate eight artificial datasets to analyze the relationships between model accuracies and dataset characteristics (non-normality, categorical feature, and dataset size). The analyses show that NABE performs significantly better than other methods on the artificial datasets with severe non-normality and large proportion of categorical features.

In the domain of cost estimation, the lessons learnt via this study are as follows:

- The non-linear based adjustment to ABE system is generally an effective approach to extend ABE's flexibility on complex datasets and improve the accuracy of ABE.

- NABE is likely to be a more accurate method than other types of ABE methods on the dataset with high degree of non-normality and large proportion of categorical features.
- On the dataset with a relatively small size, a relatively small proportion of categorical features and a moderate non-normality, NABE may not be an ideal option, since it is likely to have equal accuracy to other ABE methods and it has a more complex structure than other ABE methods.
- There are strong relationships between the successes of NABE and dataset properties (non-normality, categorical feature, and dataset size). Thus, the practitioners should be aware of the tradeoffs among datasets properties, model complexity and model accuracy, when implementing NABE.

Nevertheless, there are also some limitations of NABE. To focus on different adjustment mechanisms, we pre-determined the similarity measure and the retrieved solution function in ABE system. However, there are many options for these two components. For the similarity measures there are alternatives based on Manhattan and Minkowski distances (Mendes, et al. 2003, Huang and Chiu, 2006, Li and Ruhe, 2007), and for the retrieved solutions there are weighted mean and median (Angelis and Stamelos, 2000, Mendes, et al. 2003).

Moreover, feature selection (Kirsopp et al. 2003) and project selection (Li et al. 2009a) are important preprocessing steps of ABE method since there are often many irrelevant features and noisy projects in the software engineering datasets. The possibility of further improvement of the NABE systems also lies in the appropriate selection of relevant features and representative projects.

Furthermore, missing values often appear in the software engineering datasets. Many studies (Myrtveit et al. 2001, Strike et al. 2001, Jonsson and Wohlin 2006, Song and Shepperd 2007) have proposed different data imputation techniques to recover missing data by estimating replacement values. However, the missing values are excluded from our study. This might cause some difficulties for practitioners to apply the NABE system to the datasets with significant amount of missing values. For example during the ISBSG subset preparation, we realize that missing values cause the deletion of many projects.

Finally, the non-linear adjustment in our study is based on artificial neural networks. Other types of non-linear approximations such as Radius Basis Functions (Hardy, 1971) and Support Vector Machines (Vapnik, 1995) can also be employed as the non-linear adjustment. They may achieve better performance than ANN does, because they have fewer parameters than ANN and they have the regularization mechanism to prevent the over-fitting problem suffered by ANN.

Chapter 6 Probabilistic Analogy Based Estimation[4]

Most published research works have been focusing on improving ABE's accuracy (such as the works in Chapter 3, Chapter 4, and Chapter 5). However, due to the inherent uncertainties and complexities in cost estimation process, the accurate point estimates are often obtained with great difficulties. From the perspective of industrial engineering, it is more practical to generate probabilistic predictions. In the literature, there is still a lack of formal framework for ABE to generate probabilistic predictions. In this chapter, we first propose a probabilistic model of ABE (PABE). The prediction of PABE is obtained by integrating over the parameter *K*, the number of nearest neighbors, via Bayesian inference. In addition, PABE is tested on two well-known datasets with comparisons against other established estimation techniques. The promising results show that PABE could largely improve the point estimations of ABE and achieve quality probabilistic predictions.

6.1 Introduction

Several techniques have been proposed to improve ABE's accuracy. However, it still has been reported that ABE sometimes produces misleading

[4] This chapter is associated with the papers Li et al. 2008a and Li et al. 2008b.

results (van Koten and Gray 2006). This may be due to the inherent uncertainty in the estimation process (because a cost estimate is an assessment of a future condition and therefore the uncertainty is embedded in the estimation) and the unclear project requirement in the early stages of software life cycle. Therefore, Angelis and Stamelos (2000) pointed out that it is safer to generate probabilistic predictions such as probability distribution of the cost value or interval estimate of cost with a certain probability.

Recently, more and more researches are devoted onto probabilistic predictions. The published studies include bootstrapped ABE method (Angelis and Stamelos 2000), expert judgments (Jorgensen and Sjoberg 2003), and Bayesian networks (van Koten and Gray 2006). However, very few of these studies have proposed probabilistic model for conventional ABE. The bootstrapped ABE (Angelis and Stamelos 2000) is one important attempt. However, bootstrapping technique is a simulation based re-sampling method with high computational cost and limited interpretations. To the best of our knowledge, there is still a lack of interpretable and efficient formal probabilistic model of ABE. In other research fields, few initiatives (Holmes and Adams 2002) on probabilistic K-Nearest Neighbor Regression (KNNR) model have been taken. ABE is equivalent to KNNR in the statistics literature.

In this chapter, we present a continuous probabilistic model of ABE (PABE). Then, Bayesian inference is used to produce the probabilistic prediction by integrating over the parameter K, the number of nearest neighbors, because Bayesian inference is capable of handling missing data, learning causal relationships, combining prior knowledge and data, and avoiding over-fitting problems (Heckerman 1997).

The rest of this chapter is organized as follows. Section 6.2 presents a brief introduction to the formal model of ABE and its parameter *K*, the number of nearest neighbors. Section 6.3 describes the prior distributions of PABE model, the Bayesian inference approach, and the predictive PABE model. In section 6.4, the experiments setup for empirical validations is presented. The last section presents the results and comparisons on two object-oriented maintenance datasets.

6.2 Formal Model of Analogy Based Estimation

As pointed out by Mittas *et al.* (2008), although ABE seems to be an empirical technique, it still has the mathematical form which is known as K-Nearest Neighbor Regression in the context of statistics. Prior to the introduction of the probabilistic model of ABE, we recall ABE's formal model in Mittas *et al.* (2008)'s work.

Let $D = \{X, Y\} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\}$ be the historical dataset which contains a set of n independent historical projects, $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\} = (x_{ij})_{n \times d}$ be an $n \times d$ random matrix of project features (or cost drivers) with $\boldsymbol{x}_i$ as a vector of d project features and x_{ij} as the jth feature of the ith project, and $Y = \{y_1, y_2, \ldots, y_n\}$ be an n–dimensional random vector with y_i as the cost value of the *i*th project. Given a new project $\boldsymbol{x}$ and its cost value y unknown, the ABE's point estimate of y is the weighted sum of the cost values of $\boldsymbol{x}$'s K nearest neighbors:

$$\hat{y} = \hat{Y}(\boldsymbol{x}) = \sum_{\{i|\boldsymbol{x}_i \in N_K(\boldsymbol{x})\}} w_i(\boldsymbol{x}_i, \boldsymbol{x}) y_i \tag{6.1}$$

where $N_K(\boldsymbol{x})$ is the neighborhood set of $\boldsymbol{x}$ which comprises K nearest projects of $\boldsymbol{x}$, and $w_i(\boldsymbol{x}_i, \boldsymbol{x})$ is the weight function depending on the features of historical project $\boldsymbol{x}_i$ and new project $\boldsymbol{x}$. Usually, the summation of weights is set to be 1 for the purpose of normalization.

The number of nearest neighbors K is the key parameter of ABE. Many studies (Li and Ruhe. 2008, Mittas *et al.* 2008, Li *et al.* 2008c, Chiu and Huang 2007, Huang and Chiu 2006, Auer *et al.* 2006, Mendes *et al.* 2003, Leung 2002, Angelis and Stamelos 2000, Shepperd and Schofield 1997) have attempted to optimize this parameter; however most optimization methods are brute force empirical approaches. Table 2.3 in Chapter 2 summarizes the relevant works. It is shown that most previous studies have specified a certain range for K values and this is followed by the cross-validation procedure to select the K value with which the ABE could produce the predictions optimizing the error value on the training dataset. Moreover, three papers (Chiu and Huang 2007, Leung 2002, Auer *et al.* 2006) predefined K at fixed values, and Li and Ruhe (2008) proposed a method named 'dynamic K'. In this approach, the projects falling within a certain distance threshold (T) of the target project are treated as nearest neighbors and the number of neighbors may vary when different target projects appear. However, this method is also under the cross validation scheme. The advantage of cross validation is that it takes into account the effort data under study. Regardless of how K is selected, the predictions made by all ABE studies in table 2.3 have no probabilistic interpretations. This lack of a probabilistic interpretation in the predictions makes it difficult to incorporate ABE into the software cost estimation process where predictions are uncertain in nature.

6.3 Probabilistic Model of Analogy Based Estimation

The objective of PABE model is to obtain the marginal distribution of the cost value $p(y \mid \boldsymbol{x}, Y, X)$ given the historical dataset $D = \{X, Y\}$ and the features of project $\boldsymbol{x}$ regardless of the value of K. In this section, we first define the conditional prior distribution $p(Y \mid X, K)$ and the conditional predictive distribution $p(y \mid \boldsymbol{x}, Y, X, K)$, then utilize Bayesian inference to obtain the final marginal distribution $p(y \mid \boldsymbol{x}, Y, X)$.

6.3.1 Assumptions

Before the definition of conditional prior distribution, three fundamental assumptions of PABE are introduced as follows:

1) Given the historical project dataset D, the cost value y of project $\boldsymbol{x}$ is a normally distributed independent variable with mean μ and variance σ^2.

2) The mean value μ equals to the output of the conventional ABE model in (6.1).

3) The variance σ^2 is related to the summation of the similarities between project $\boldsymbol{x}$ and its K nearest neighbors.

Assumption 1) appears invalid in the real world applications, since the real cost values must be positive and are often with extreme outliers. However, the cost value can be transformed to normal distribution. Experience and theory have shown that the logarithmic and square root transformation can effectively produce normally distributed quantities from non-normal distributions (Angelis and Stamelos 2000, Jeffery *et al.* 2000, Mittas *et al.* 2008).

Assumption 2) requires that the expectation of PABE should be point ABE estimate in (6.1), so that we can obtain μ as:

$$\mu = E(Y \mid D) = E\Big(\sum_{\{i \mid x_i \in N_K(x)\}} w_i(\boldsymbol{x}_i, \boldsymbol{x}) y_i + \varepsilon\Big) = \sum_{\{i \mid x_i \in N_K(x)\}} w_i(\boldsymbol{x}_i, \boldsymbol{x}) y_i \tag{6.2}$$

where the weight $w_i(\boldsymbol{x}_i, \boldsymbol{x})$ is defined as:

$$w_i(\boldsymbol{x}_i, \boldsymbol{x}) = \frac{s_i}{\sum_{\{i \mid x_i \in N_K(x)\}} s_i} \tag{6.3}$$

where s_i is the similarity between project $\boldsymbol{x}$ and its ith nearest project and $N_K(\boldsymbol{x})$ is the neighborhood set of $\boldsymbol{x}$. Recall from the formula of similarity measure in (2.11), s_i is defined as follows:

$$s_i = \frac{1}{c + d_i} \tag{6.4}$$

where d_i is the Euclidean distance between project $\boldsymbol{x}$ and its ith nearest project, and c is a small constant to avoid the situation $d_i = 0$. We set $c = 0.0001$ in this study.

Assumption 3) reflects the intuition that the estimations with higher total similarity $s = \sum_{x_i \in N_K} s_i$ acquire higher precision. Assumption 3) is reminiscent of the definition of precision matrix in Gaussian Markov Random Field (Ferreira and Victor 2007).

6.3.2 Conditional Distributions

By assumption 1), the cost value y_i is independent from each other. Given the feature matrix X and parameter K, the conditional distribution of Y is described as below:

$$p(Y \mid X, K) \sim \prod_{i=1}^{n} N\left(\sum_{x_j \in (n-1)_K} \frac{s_{ij}}{s_{ii}} y_j , \frac{1}{1 + s_{ii}} \right) \tag{6.5}$$

where n denotes the total number of projects in the historical dataset, $(n - 1)_K$ represents the K sized neighborhood system in the set of $(n - 1)$ projects excluding project $\mathbf{x}_i$, s_{ij} is the similarity between project $\mathbf{x}_i$, and $\mathbf{x}_j$, $s_{ii} = \sum_{\mathbf{x}_j \in (N-1)_K} s_{ij}$ is the total similarity of project $\mathbf{x}_i$. Under this circumstance, the probability of y_i is conditioned on only those K nearest projects.

Given the features of new project $\mathbf{x}$, the conditional predictive distribution of its cost value y is:

$$p(y \mid \mathbf{x}, Y, X, K) \sim N\left(\sum_{x_j \in N_K} \frac{s_j}{s} y_j, \frac{1}{1+s} \right) \tag{6.6}$$

where $s = \sum_{\mathbf{x}_j \in N_K} s_j$ is the total similarity between project $\mathbf{x}$ and its K nearest neighbors.

6.3.3 Predictive Model and Bayesian Inference

Our goal is to obtain the marginal predictive model $p(y \mid \mathbf{x}, Y, X)$ regardless of the value of K. It can be obtained by integrating the joint predictive distribution (6.6) over the space of parameter K:

$$\begin{aligned} p(y \mid \mathbf{x}, Y, X) &= \int p(y \mid \mathbf{x}, Y, X, K) p(K \mid \mathbf{x}, Y, X) dK \\ &= \sum_K p(y \mid \mathbf{x}, Y, X, K) p(K \mid Y, X) \end{aligned} \tag{6.7}$$

where the conditional predictive model $p(y \mid \mathbf{x}, Y, X, K)$ is given in (6.6), and the posterior distribution $p(K \mid Y, X)$ can be solved by Bayesian inference. The Bayesian inference often involves three steps: 1. assign a prior $p(K)$ to the unknown parameter K; 2. define the likelihood $p(Y \mid X, K)$ of observing Y given K; 3. determine the posterior $p(K \mid Y, X)$ of K.

The prior distribution of K is considered first. It is important to choose an appropriate prior distribution which results in computationally tractable posterior distribution. In our study, the prior of the parameter K is assumed to be uniformly distributed, since we have little prior knowledge about the likely values of K, and the uniform distribution is a maximum entropy distribution.

$$p(K) = U\{1, \ldots, K_{\max}\} \tag{6.8}$$

where U denotes the uniform distribution. The distribution $p(Y \mid X, K)$ is given in (6.5). With the (6.5) and (6.8), the posterior distribution of K can be rewritten as:

$$p(K \mid Y, X) = p(Y \mid X, K)\, p(K) \frac{p(X \mid K)}{p(Y, X)} \tag{6.9}$$

By definition, the occurrences of the observed features X are independent from the value of K, and therefore $p(X \mid K)$ can be regarded as some constant. Then (6.9) can be rewritten as follows:

$$\begin{aligned} p(K \mid Y, X) &= \frac{p(Y \mid X, K) p(K)}{\int p(Y, X, K) dK} p(X \mid K) \\ &= \frac{p(Y \mid X, K) p(K)}{\int p(Y \mid X, K) p(X \mid K) p(K) dK} p(X \mid K) \\ &= \frac{p(Y \mid X, K) p(K)}{\int p(Y \mid X, K) p(K) dK} = \frac{p(Y \mid X, K) p(K)}{\sum_{K=1}^{K_{\max}} p(Y \mid X, K) p(K)} \end{aligned} \tag{6.10}$$

Since the prior of K is uniformly distributed and $p(K) = 1/K_{\max}$, then the posterior in (6.10) can be rewritten as:

$$p(K \mid Y, X) = \frac{p(Y \mid X, K)}{\sum_{K=1}^{K_{\max}} p(Y \mid X, K)} \tag{6.11}$$

where $p(Y \mid X, K)$ is the conditional distribution of Y defined in (6.5). The value of $p(Y \mid X, K)$ can be computed by substituting the historical projects features and effort values into its equation. To simplify the notations, let $p(Y \mid X, K) \sim \prod_{t=1}^{n} N\left(\sum_{x_j \in (N-1)_K} \frac{s_{ij}}{s_{ii}} y_j, \frac{1}{s_{ii}+1} \right) = Z_K$, so that the final predictive model has the following form:

$$
\begin{aligned}
p(y \mid \boldsymbol{x}, Y, X) &= \int p(y, K \mid \boldsymbol{x}, Y, X) dK \\
&= \sum_k p(y \mid \boldsymbol{x}, Y, X, K) p(K \mid Y, X) \\
&= \sum_{k=1}^{K_{\max}} p(y \mid \boldsymbol{x}, Y, X, K) \frac{p(Y \mid X, K)}{\sum_{K=1}^{K_{\max}} p(Y \mid X, K)} \\
&\sim \sum_{K=1}^{K_{\max}} \left[N\left(\sum_{x_j \in N_K} \frac{s_j}{s} y_j , \frac{1}{s+1} \right) \cdot \frac{Z_K}{\sum_{K=1}^{K_{\max}} Z_K} \right]
\end{aligned}
\qquad (6.12)
$$

It is noted that the predictive PABE model is in fact the weighted summation of a series of normal distributions. Therefore, the resulting distribution is also normally distributed and the mean and variance of this distribution are:

$$
\mu_{K\max} = \sum_{K=1}^{K_{\max}} \left(f_K \sum_{x_j \in N_K} \frac{s_j}{s} y_j \right) \qquad (6.13)
$$

$$
\sigma^2_{K\max} = \sum_{K=1}^{K_{\max}} \left[f_K^2 \frac{1}{s+1} \right] \qquad (6.14)
$$

where $f_K = \frac{Z_K}{\sum_{K=1}^{K_{\max}} Z_K}$ represents the proportion of the probability of observing Y with K neighbors, and $s = \sum_{\boldsymbol{x}_j \in N_K} s_j$ is changing with different K values. Comparing with the point ABE model in (6.1), the expectation of PABE model in (6.13) is further adjusted by a sequence of weights f_K which reflects the likelihood of historical data with different K values. In applications, $\mu_{K\max}$ and $\sigma^2_{K\max}$ together determine the final predictive model.

From the presentations above, we obtain the predictive PABE by integrating over the parameter K instead of relying on some optimal K values, because optimization often fails to take into account the inherent uncertainty in parameters. There is no 'true' value for the parameter which can be found by optimization. However, there is a range of possible values for the parameter, each with some associated density (Denison et al. 2002).

6.3.4 Implementation Procedure of Probabilistic Analogy Based Estimation

The detailed implementation procedure of PABE to predict the cost value y of a new project x is presented as follows:

1. Prepare the historical data set $D = \{X, Y\}$, where X is the project feature matrix and Y is the vector of project efforts. Take the necessary transformation (logarithm or square root) to transform Y to normal distribution. Set $K_{\max}$ equal to 10.
2. Calculate the point estimate $\sum_{x_j \in N_K} \frac{s_i}{s} y_j$ and the total similarity $s = \sum_{x_j \in N_K} s_j$ of the new project $\boldsymbol{x}$ for each K value from 1 to 10. The similarity is calculated by (6.4). In all, a vector of ten cost estimates and a vector of ten total similarities are obtained.

3. Calculate the likelihood of observing Y, $P(Y \mid X, K)$ for each K value from 1 to 10 by formula (6.5). At this step, a vector of ten likelihood values is obtained.
4. Calculate the probability distribution y by specifying $\mu_{K\max}$ and $\sigma^2_{K\max}$ using the formula in (6.13) and (6.14).
5. Convert the point prediction $\mu_{K\max}$ and interval estimation $[\mu_{K\max} \pm Z_{\alpha/2} \cdot \sigma_{K\max}]$ to the final predictions by exponential or square transformation.

6.4 Experiment Design

In this section, we evaluate PABE on two real world datasets with the comparisons to other estimation techniques. The datasets are described first. Then the prediction accuracy measures of point prediction and interval prediction are introduced. After that, the cross-validation scheme is presented. Lastly, other comparative methods and the detailed experiment procedures are described.

6.4.1 Datasets

For the purpose of comparing PABE to the published software maintenance effort estimation methods, we select the two well known objective oriented software maintainability datasets by Li and Henry (1993). These datasets have been frequently used by recent studies to evaluate their methods (Zhou and Leung 2007, van Koten and Gray 2006, Thwin and Quah 2005). The first data set, UIMS, contains 39 classes collected from a User Interface Management System. The second data set, QUES, contains 71 classes from a QUality Evaluation System. Both UIMS and QUES datasets consist of 11 metrics: 9 object-oriented metrics, one traditional size metric, and one maintainability metric. Among the object-oriented metrics, WMC, DIT, RFC, NOC, and LCOM are proposed by Chidamber and Kemerer (1994), and MPC, DAC, NOM and SIZE2 are proposed by Li and Henry (1993). SIZE1 is the traditional lines of code size metric. Maintainability is measured with the CHANGE metric by counting the number of lines in the code that were changed per class during a 3-year maintenance period. Table B.9 provides the description of each metric.

The descriptive statistics of the UIMS and QUES datasets are shown in table B.10 and table B.11 respectively. As Briand *et al.* (2000) pointed out, metrics that vary little are not likely to be useful predictors and only the metrics with more than five non-zero values are recommended for experiments. In table B.10, most metrics of UIMS dataset show large variance except DIT. For DIT, the number of its non-zero values is larger than five. Thus, all metrics of UIMS data set are used in experiments. From table B.11, it is seen that NOC has only zero values. Therefore, the metric NOC is removed from QUES dataset for experiments.

Table 6.1 shows the Spearman's rank correlations between maintainability and the OO metrics on UIMS and QUES datasets. There is a significant correlation between CHANGE and the OO metrics. However, table 6.1 also shows that the correlations in the UIMS dataset are different from the correlations in the QUES dataset. In addition, table B.10 and table B.11 show that the characteristics of the UIMS dataset are different from the QUES dataset. Thus, the UIMS and QUES datasets are regarded as heterogeneous.

Table 6.1: Correlations between CHANGE and OO metrics

Metric	Spearman's correlation coefficient	
	UIMS dataset	QUES dataset
DIT	-0.10	-0.04
NOC	0.31	NA
MPC	0.69*	0.55*
RFC	0.63*	0.38*
LCOM	0.76*	-0.05
DAC	0.48*	-0.19
WMC	0.73*	0.08
NOM	0.62*	0.05
SIZE1	0.76*	0.62*
SIZE2	0.57*	-0.01

Correlation is significant at the 0.01 level (2-tailed)

6.4.2 Prediction Accuracy

Since PABE can produce both point and probabilistic predictions, the performance metrics for these two kinds of predictions are introduced in this section.

Point prediction evaluation

To measure the accuracies of effort estimation methods, the selection of accuracy metrics is crucial. The magnitude of relative error (MRE) is the de facto error metric in software effort estimation literature. Based on it, Mean Magnitude of Relative Error (MMRE), Max Magnitude of Relative Error (MaxMRE), Median Magnitude of Relative Error (MdMRE) and PREDiction at level k PRED(k) are proposed to describe different aspects of MRE. In this study, q is set to be 0.25 and 0.30 since they are commonly used in the cost estimation literature (Lucia *et al.* 2005, Kitchenham *et al.* 2002). The MaxMRE measures the maximum relative discrepancy which is the maximum error relative to the actual value in the prediction (van Koten and Gray 2006). The PRED identifies the estimations that are generally accurate, while MMRE is a biased and not always reliable as a performance metric.

Probabilistic Prediction Evaluation

The probabilistic predictions can be easily transformed into interval predictions with a certain probability. Evaluating prediction intervals is different from evaluating point estimates. A point estimate can be compared with the actual value, while an interval prediction has no corresponding actual value. Instead, the ‘hit rate’ (Jørgensen and Sjøberg 2003), which calculates the proportion of the projects with the actual cost falling into the prediction interval, is considered as the accuracy measure in our study.

$$HitRate = \frac{1}{n}\sum_{i=1}^{n} h_i$$
$$h_i = \begin{cases} 1, & \min_i \le Act_i \le \max_i \\ 0, & Act_i > \max_i \; or \min_i > Act_i \end{cases} \tag{6.15}$$

where min$_i$ and max$_i$ are the minimum and maximum values of the prediction interval for project *i*, Act$_i$ is the actual cost of project *i* and *n* is the total number of projects being estimated.

In addition, the efficient use of uncertainty information means that the prediction interval can be narrower without losing the correspondence between confidence level and hit rate. A measure able to compare the interval prediction's efficiency is the median of the relative widths of the prediction intervals (Jørgensen and Sjøberg 2003). The width of a prediction interval is defined as:

$$PIWidth = \frac{Maximum\ effort - Minmum\ effort}{Predicted\ effort} \tag{6.16}$$

Cross Validation

For the purpose of validation, the jack-knife validation schemes is utilized in this study. The jackknife method which is also known as leave-one-out cross validation (LOOCV) is a useful tool to obtain nearly unbiased estimators of prediction error. In this approach, at each stage a project is removed from the historical dataset for testing and the remaining projects are used as the training set at each stage. This procedure is repeated N times (N is the number of projects in historical dataset) and then the accuracies across all projects are aggregated. The reasons to choose jack-knife approach are: 1) jack-knife validation is a widely used variant of v-fold cross-validation, 2) it is closer to a real world situation than k-cross validation ($k < n$) (Myrtveit *et al.* 2005), 3) unlike k-fold cross validation ($k < n$), the jack-knife validation is deterministic, i.e. no sampling is involved, 4) it ensures the largest amount of data for training which presumably increases the chance of getting more accurate predictions (Witten and Frank 2000).

Estimation Methods

Two types of ABE models are included in our experiments. The first model is the proposed PABE. The second model is conventional ABE (CABE) with the parameter K optimized by cross-validation. The specified range of K values is from 1 to 10. To eliminate the impacts from different factors, all ABE based methods are implemented with the similarity measure fixed to Euclidean based similarity.

For a more comprehensive evaluation of PABE, we also compare it with other popular machine learning methods including Stepwise Regression SWR (Mendes *et al.* 2003), Artificial Neural Network ANN (Heiat 2002), Classification and Regression Trees CART (Pickard *et al.* 2001). The best variants of these methods are obtained by tuning their parameters on the training datasets.

6.4.3 Experiment Procedure

The following procedures are taken to validate the proposed PABE model with comparisons against other methods on each dataset.

- The PABE model is implemented by jack-knife scheme with the ABE similarity measure fixing to Euclidean distance. The MREs and residuals of its point prediction, and the HitRate and PIwidth of its probabilistic predictions, across all test projects are computed.
- The conventional ABE, SWR, ANN, and CART are trained and tested by jack-knife validation. The best variants of these methods on the training sets are used to predict the testing projects. The MREs and Absolute residuals of their prediction are collected.
- The comparative results on the MRE based error metrics are analyzed and the Wilcoxon signed-rank tests are performed to identify the significance of difference in absolute residual values and MRE values of all methods.
- The Bootstrapped conventional ABE (BABE) is performed and its interval predictions are compared with PABE.

6.5 Results

In this section, the results and comparisons on each dataset are presented according to the experiment steps mentioned in section 6.4.3.

6.5.1 Results on UIMS Dataset

Table 6.2 presents the point prediction accuracies obtained by PABE, CABE, SWR, ANN, and CART on the UIMS dataset with jack knife validation. The table shows that PABE achieves the lowest MMRE and MdMRE, the highest PRED(25) and PRED(30), and the second lowest MaxMRE value among all methods. The results indicate that PABE performs generally better than other methods under MRE based error metrics except its maximum MRE value is larger than that of CABE.

Table 6.2: Point prediction accuracy on UIMS dataset

Methods	MaxMRE	MMRE	PRED(25)	PRED(30)	MdMRE
PABE	3.93	0.56	0.46	0.49	0.31
CABE	2.95	0.74	0.15	0.18	0.64
SWR	9.97	2.13	0.28	0.33	0.95
ANN	16.31	2.45	0.18	0.23	0.75
CART	14.26	2.48	0.26	0.36	0.82

To further analyze the performances, we draw out the box plots of MRE values and absolute residuals of all methods in fig 6.1 because absolute residuals are less vulnerable to bias than the asymmetric MRE values (Stensrud *et al.* 2003). For the MRE boxplots, PABE has the lowest median line and least outliers, but its box width and whiskers are slightly larger than those of CABE. For the absolute residual boxplots, PABE has the narrowest box and least outliers, but its median line is close to that of CABE and its whiskers are larger than those of CART. In all, the boxplots does not provide a clear conclusion on whether PABE is significantly better than other methods. It is also revealed that the distributions of both MREs and absolute residuals are heavily skewed. This implies that the standard *t*-test is no longer valid for the statistical comparisons. Therefore, the assumption-free Wilcoxon signed-rank tests are performed for the significance of differences.

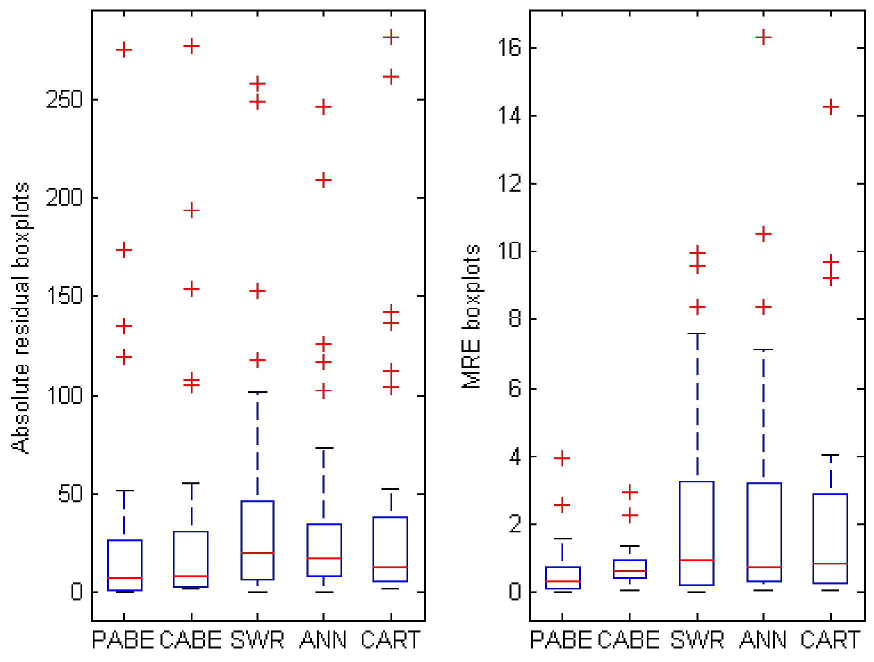

Figure 6.1: Boxplots of Absolute residuals and MREs on UIMS dataset

Table 6.3 presents the Z statistic and *p*-value of the two-tailed Wilcoxon signed-rank test for Absolute Residual (AR) and MRE values of the PABE vs. CABE, SWR, ANN, and CART paired comparisons. It is shown that PABE is significantly better than all other methods on MRE criterion and PABE is better than CABE and ANN under AR criterion with the *p*-values less than 0.05. Though the *p*-values of PABE vs. SWR and PABE vs. CART on AR criterion are greater than 0.05 which is the most widely adopted threshold, they are smaller than or equal to 0.1 which is another commonly used threshold for significant tests (Jørgensen and Sjøberg 2003).

Table 6.3: Wilcoxon signed-rank test on UIMS dataset

Methods		CABE		SWR		ANN		CART	
		AR	MRE	AR	MRE	AR	MRE	AR	MRE
PABE	*Z*-value	2.15[a]	2.36[a]	1.66[a]	2.47[a]	2.02[a]	3.41[a]	1.73[a]	2.33[a]
	p-value	0.03	0.02	0.10	0.01	0.04	0.00	0.08	0.02

[a] $T_+ < T_-$.

For the interval based predictions, PABE is compared against the well established bootstrapping ABE (BABE) proposed by Angelis and Stamelos (2000). The results are summarized in table 6.4. In terms of HitRate, PABE achieves higher hit rate than BABE. With respect to the Median PIwidth, though PABE's intervals are wider than those of BABE, they are still in the reasonable range (smaller than 3). Another important advantage of the PABE is its computational efficiency. The time in seconds needed to train the models in the jack-knife validation is recorded in the last row of table 6.4. It is shown that PABE is more than 10 times faster than BABE. Fig 6.2 presents the actual efforts of the same data set along with the 95% confidence intervals by PABE and BABE. For better interpretation, in fig 6.2 we present all the lower and all the upper bounds connected with a dashed line forming a 95% *confidence zone*. As we can see from the figures, the PABE generally has wider confidence intervals on UIMS dataset compared to those of BABE.

Table 6.4: Results of interval prediction at 95% confidence level

Quality metrics	PABE	BABE
HitRate	0.67	0.56
MPIwidth	2.82	1.50
Time used (s)	29	428

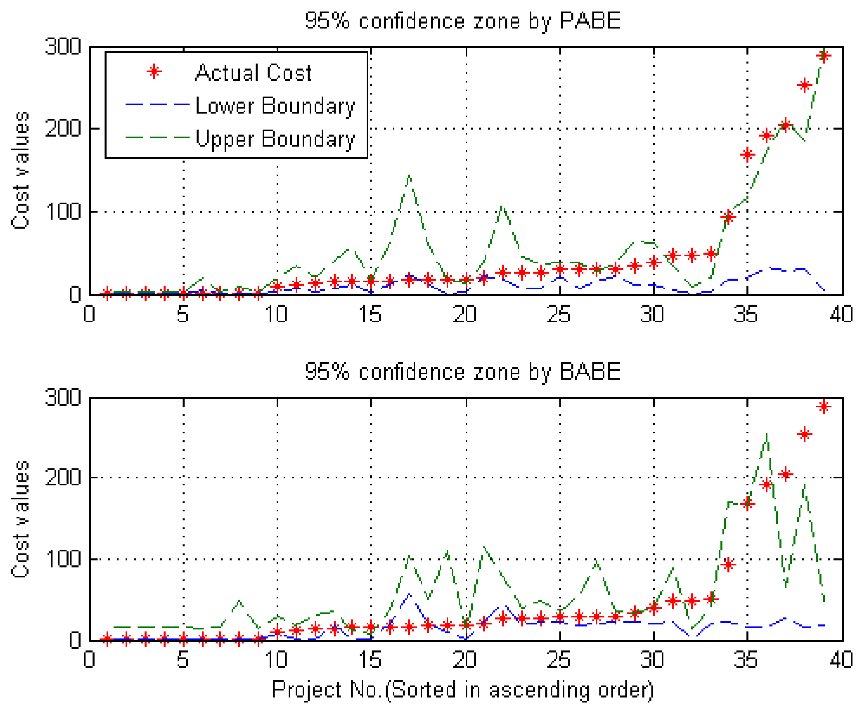

Figure 6.2: Confidence zones on UIMS dataset

6.5.2 Results on QUES Dataset

The point prediction accuracies obtained by PABE, CABE, SWR, ANN, and CART on the QUES dataset are presented in table 6.5.

Table 6.5: Point prediction accuracy on QUES dataset

Methods	MaxMRE	MMRE	PRED(25)	PRED(30)	MdMRE
PABE	1.25	0.27	0.59	0.66	0.17
CABE	1.43	0.31	0.58	0.61	0.19
SWR	1.98	0.37	0.44	0.54	0.28
ANN	2.31	0.42	0.42	0.51	0.29
CART	3.13	0.57	0.38	0.39	0.34

It is shown that PABE achieves the lowest MaxMRE, MMRE and MdMRE, and the highest PRED(25) and PRED(30). Different from the results on UIMS dataset, CABE's error metric values seem very close to those of PABE.

For a further analysis, the box plots of MRE values and absolute residuals of all methods are illustrated in fig 6.3. For the MRE boxplots, PABE has the narrowest box and whiskers, but its median line is very close to that of CABE and it has more outliers than CABE and ANN. For the absolute residual boxplots, PABE has the lowest median line, the narrowest box and whiskers, but it has the highest number of outliers. In all, the boxplots do not provide a clear conclusion on whether PABE is better than other methods. Thus, the Wilcoxon signed-rank tests are performed for the significance of differences.

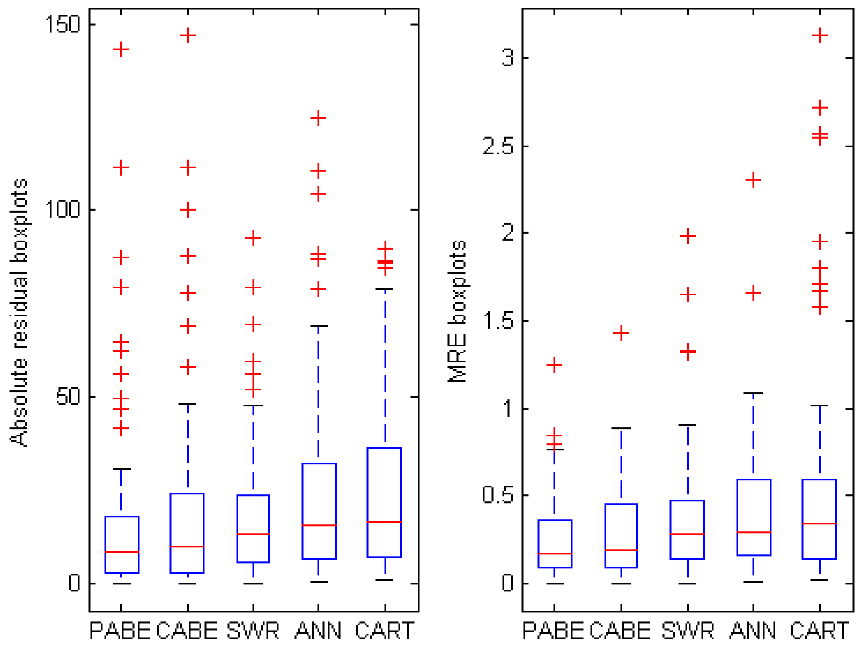

Figure 6.3: Boxplots of Absolute residuals and MREs on QUES dataset

Table 6.6 presents the Z statistic and *p*-value of the two-tailed Wilcoxon signed-rank test for Absolute Residual (AR) and MRE values of the PABE vs. CABE, SWR, ANN, and CART paired comparisons. It is shown that PABE is significantly better than all other methods on both AR and MRE criteria with the *p*-values less than 0.05.

Table 6.6: Wilcoxon signed-rank test on QUES dataset

Methods		CABE		SWR		ANN		CART	
		AR	MRE	AR	MRE	AR	MRE	AR	MRE
PABE	*Z*-value	2.67[a]	2.99[a]	2.02[a]	2.54[a]	3.23[a]	3.41[a]	3.41[a]	4.01[a]
	p-value	0.01	0.00	0.04	0.01	0.00	0.00	0.00	0.00

[a] $T_+ < T_-$.

Table 6.7: Results of interval prediction at 95% confidence level

Quality metrics	PABE	BABE
HitRate	0.94	0.74
MPIwidth	2.42	0.76
Time used (s)	202	1248

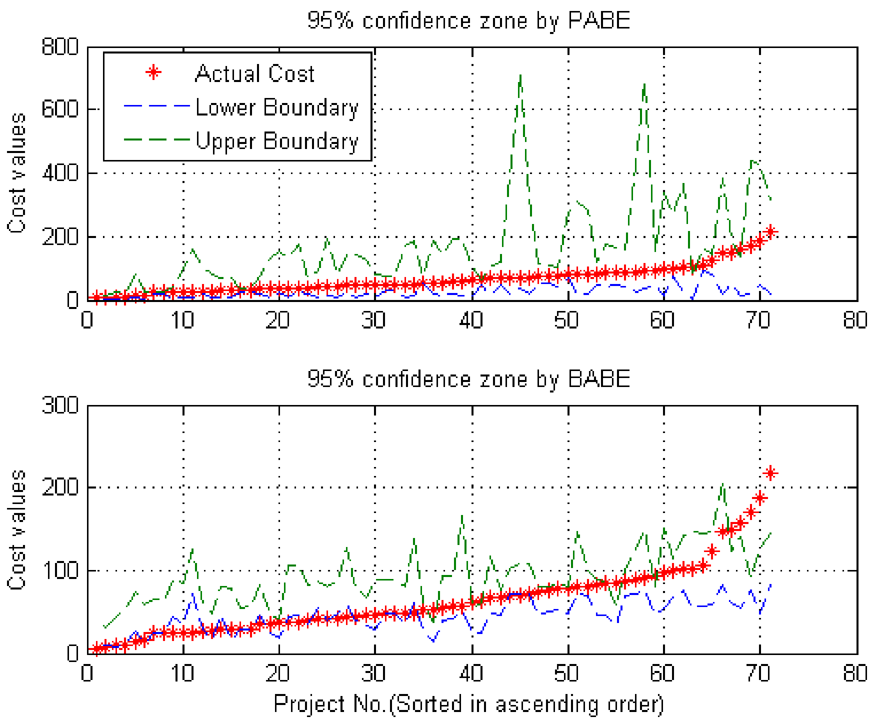

Figure 6.4: Confidence zones on QUES dataset.

For the interval based predictions, similar to UIMS dataset, PABE is compared against bootstrapping ABE (BABE). The results are summarized in table 6.7. PABE achieves higher HitRate than BABE and its HitRate is close to the corresponding confidence level (95%). As to the Median PIwidth, PABE's intervals are wider than those of BABE, but they are still in the reasonable range (median smaller than 2.5). Table 6.7 also presents the computational efficiency of each method. It is observed that PABE is more than 6 times faster than BABE on QUES dataset. Fig 6.4 presents the actual costs of the same data set along with the 95% confidence intervals generated by PABE and BABE. As we can observe from fig 6.4, PABE's intervals cover most actual cost values.

Chapter 7
Conclusions and Future Works

Aimed at improving analogy based software cost estimation, this thesis is composed of several studies on the components of analogy based method. The research works are grouped into four chapters: chapter 3 summarizes the works on mutual information based feature selection technique for similarity function; chapter 4 presents the research on genetic algorithm based project selection method for historical database; chapter 5 presents the work on non-linear adjustment to solution function; chapter 6 presents the probabilistic model of analogy based estimation which is focused on the number of nearest neighbors.

Research in chapters 3 to 5 aims to enhance the analogy based estimation's capability to achieve more accurate results. For instance, in chapter 5 the adjustment mechanism has been largely improved for a more accurate analogy based method. Efficiency is another important aspect of estimation performance. In chapter 3, the study on refining the historical dataset has achieved a significant reduction of unnecessary projects. Consequently, the efficiency of analogy based system is largely improved. Moreover, in chapter 6 the study on probabilistic model leads to a more robust and reliable analogy based method. This study could enhance the system's capability to deal with a wider range of complicated situations such as missing values and ambiguous inputs.

From the perspectives of software engineering, these works lead to an in depth knowledge of analogy based cost estimation with significant enhancement of ABE's accuracy, efficiency and robustness. From the perspectives of industrial engineering, these works can be regarded as successful applications of IE methodologies, such as optimization and probabilistic modeling. To highlight the contributions and the feasible extensions of our research, the following paragraphs summarize the conclusions and the possible future works of chapters 3 to 6 individually.

In chapter 3, the feature selection for similarity measure is investigated. Mutual Information based hybrid wrapper and filter feature selection scheme (MIABE) is proposed to improve the efficiency and the interpretation of the existing feature selectors. The results suggested that MIABE could achieve better predictions on testing datasets (generalization) though MIABE did not perform very well on fitting the training datasets. In addition, MIABE can obtain more meaningful features which can be explained by mutual information diagram. Another important finding is that MI based feature selection is more efficient than the wrappers, especially when there are more features in the dataset.

However, there are limitations in this study. First, comparisons with the wrapper methods are restricted to fixed parameter settings. Based on the fixed parameter setting, it is difficult to conclude that MI based feature selection could achieve equally good results under other conditions. Therefore, sensitivity analysis i.e. how prediction performance is affected by varying parameters, is worth to investigate for future work. In addition, only two real world datasets were used for experiments in this study. Future work could include more real world datasets (such as ISBSG dataset) for the validation of MIABE.

Chapter 4 focuses on the subset selection of historical database. We introduce the powerful genetic algorithm to perform the optimization of project selection as well as the simultaneous optimization of feature weights and project selection. The promising results clearly indicate that project selection can improve prediction accuracies and reduce the computation complexity.

One of the major limitations of chapter 4 is the dataset. The experiments in chapter 4 were performed on two relatively aged but frequently used datasets. The projects in these two datasets were developed by the traditional waterfall approach. However, a large number of recent software projects are developed by new types of software development models (such as spiral model and agile model) which often have new types of project features (such as percentage of reuse) different from the waterfall approach. Therefore, new datasets should be considered for the validation of our method to generate more meaningful conclusions for the contemporary cost estimation practices. Moreover, our current analogy based method is intolerant to missing feature values. In future work, the data imputation techniques can be taken into considerations to fill the missing information in the historical project databases.

In chapter 5, adjustment to the solution function is studied. A non-linear adjustment mechanism with learning ability and incorporating categorical features is proposed. The results show that NABE is generally an effective approach to extend ABE's flexibility on complex datasets and to improve the accuracy of ABE on complex datasets. NABE is likely to be a more accurate method than other types of ABE methods on the dataset with high degree of non-normality and large proportion of categorical features.

Nevertheless, there are some limitations of NABE. The similarity measure and solution function are pre-determined in this study. Further studies can be designed to systematically investigate the influences of similarity measure and solution function. The sensitivity analysis on these components can be conducted as well. Moreover, additional real world datasets and additional dataset characteristics can be explored to enhance the external validity of the current research. Thirdly, other types of non-linear approximators, such as RBF and SVM, could be considered as the adjustment.

In chapter 6, the probabilistic model of ABE is proposed and validated. We first propose an analytical probabilistic framework for ABE (PABE) which accounts for uncertainty that is often ignored in the conventional ABE method. The predictive model is generated by integrating over the parameter K via Bayesian inference. The results show that PABE could produce promising results. For the point estimation, it is more accurate than conventional ABE, stepwise regression, artificial neural networks and classification and regression trees. For the interval prediction, PABE generates higher hit rates than BABE with prediction intervals' width in a reasonable range.

However, there are some limitations for PABE. The similarity measures and retrieved solutions are fixed in this study. Future works can be done to investigate the impacts from these parameters. Exploring more data characteristics and including more data sets for experiments could enhance the external validity of the findings. Thirdly, other types of effort value distributions could be incorporated into future studies. Moreover, PABE model assumes that software projects are independent from each other. However, in real world applications, many projects are accomplished in similar environment, and hence it is highly possible that some projects are related to each other. How to incorporate the interactions between projects remains to be one important issue of future research.

Bibliography

Abran, A., Robillard, P. N. 1996, Function points analysis: an empirical study of its measurement processes, *IEEE Transactions on Software Engineering*, 22(12), 895-910.

Adriano L. I. 2006. Estimation of software project effort with support vector regression. *Neurocomputing*, 69(13-15), 1749 – 1753.

Agrawal M., Chari K. 2007. Software effort, quality, and cycle time: A study of CMM level 5 projects. *IEEE Transactions on Software Engineering*, 33(3), 145-156.

Aguilar-Ruiz J. S., Ramos I., Riquelme J. C., Toro M. 2001. An evolutionary approach to estimating software development projects. *Information and Software Technology*, 43(14), 875-882.

Ahmeda M. A., Saliub M. O., AlGhamdia J. 2005. Adaptive fuzzy logic-based framework for software development effort prediction. *Information and Software Technology*, 47(1), 31–48.

Ahn Y., Suh J., Kim S., Kim H.. 2003. The software maintenance project effort estimation model based on function points. *Journal of Software Maintenance and Evolution: Research and Practice*, 15(2), 71-85.

Albrecht A.J., Gaffney J. 1983. Software function, source lines of code, and development effort prediction, *IEEE Transactions on Software Engineering* 9(6), 639–648.

Albrecht A.J. 1979. Measuring application development productivity. *Proceedings of the Joint SHARE/GUIDE and IBM Application Development Symposium*, 83–92.

Andolfi M., Costamanga M., Paschetta E., Rosenga G. 1996. A multicriteria-based methodology for the evaluation of software cost estimation models and tools, *Conference on Software Measurement and Management*, 239-248.

Angelis L., Stamelos I., Morisio M. 2001. Building a software cost estimation model based on categorical data, *Proceedings of the 7th IEEE International Software Metrics Symposium*, 4–15.

Angelis L., Stamelos I. 2000. A simulation tool for efficient analogy based cost estimation. *Empirical Software Engineering*, 5(1), 35–68.

Abran A., Silva I., Primera L. 2002 Field studies using functional size measurement in building estimation models for software maintenance. *Journal of Software Maintenance and Evolution Research and Practice*, 14(1), 31-64.

Armstrong J.S. 2001. The forecasting dictionary, in: *Principles of Forecasting: A Handbook for Researchers and Practitioners*, Kluwer, Boston.

Auer M., Trendowicz A., Graser B., Haunschmid E., Biffl S. 2006. Optimal project feature weights in analogy-based cost estimation: Improvement and limitations. *IEEE Transactions on Software Engineering*, 32(2), 83-92.

Barcelos Tronto I.F., Silvaa J. D. S., Sant'Anna N. 2007. An investigation of artificial neural networks based prediction systems in software project management. *Journal of Systems and Software*, 81(3), 356-367.

Benediktsson O., Dalcher D., Reed K. 2003. COCOMO-based effort estimation for iterative and incremental software development. *Software Quality Journal*, 11(4), 265-281

Blattberg R.C., Hoch S.J., 1990. Database models and managerial intuition: 50% model + 50% manager. *Management Science*, 36(8), 887–899.

Bocco M. G., Moody D. L., Piattini M. 2005 Assessing the capability of internal metrics as early indicators of maintenance effort through experimentation. *Journal of Software Maintenance and Evolution Research and Practice*, 17(3), 225-246.

Boehm B. 1981. *Software Engineering Economics*, Prentice-Hall, Englewood Cliffs, NJ.

Boehm B., Wolverton R. W. 1980. Software cost modeling: Some lessons learned. *Journal of Systems and Software*, 1(3), 195-201.

Boehm B., Clark B., Horowitz E., Westland C. 1995. Cost models for future software life cycle processes: COCOMO 2.0. *Annals of Software Engineering*, 1(1), 57-94.

Boehm B., Abts C., Chulani S. 2000. Software development cost estimation approaches – A survey. *Annals of Software Engineering* 10(1-4), 177–205.

Briand L.C., El-Emam K., Bomarius F. 1998. COBRA: A hybrid method for software cost estimation, benchmarking and risk assessment. *Proceedings of 20th International Conference on Software Engineering*. 390-399.

Briand L.C., El-Emam K., Maxwell K., Surmann D., Wieczorek I. 1999. An assessment and comparison of common cost software project estimation methods. *Proceedings of 21st International Conference Software Engineering*. 313-322.

Briand L.C., Wieczorek I. 2002. Resource Estimation in Software Engineering. *International Software Engineering Research Network, Technical Report*.

Briand, L.C., Wust, J., Daly, J.W., Porter, D.V., 2000. Exploring the relationships between design measures and software quality in object-oriented systems. *Journal of Systems and Software*, 51(3), 245–273.

Brieman L., Friedman J., Olshen R., Stone C. 1984. *Classification and Regression Trees.* Belmont:Wadsworth Inc.

Burgess C. J., Lefley M. 2001. Can genetic programming improve software effort estimation? A comparative evaluation. *Information and Software Technology*, 43(14), 863-873.

Calzolari F., Tonella P., Antoniol G. 2001 Maintenance and testing effort modeled by linear and nonlinear dynamic systems. *Information and Software Technology*, 43(8), 477-486.

Chidamber S.R., Kemerer C.F. 1994. A metrics suite for Object-Oriented design. *IEEE Transactions on Software Engineering*, 20(6), 476-493.

Chiu N. H., Huang S. J. 2007. The adjusted analogy-based software effort estimation based on similarity distances. *Journal of Systems and Software*. 80(4), 628-640.

Chulani S., Boehm B., Steece B. 1999. Bayesian analysis of empirical software engineering cost models. *IEEE Transactions on Software Engineering*, 25(4), 573-583.

Conte S., Dunsmore H., Shen V.Y. 1986. *Software Engineering Metrics and Models.* Menlo Park, Calif.: Benjamin Cummings.

Costagliola G., Ferrucci F., Tortora G., Vitiello G. 2005. Class point: An approach for the size estimation of object-oriented systems. *IEEE Transactions on Software Engineering*, 31(1), 52-74.

Dillibabu R., Krishnaiah K. 2005. Cost estimation of a software product using COCOMO II.2000 model - a case study. *International Journal of Project Management*. 23(4), 297 – 307.

Dri A., Abran A., Mbarki S. 2006. An experiment on the design of radial basis function neural networks for software cost estimation. *International Conference on Information & Communication Technologies: from Theory to Applications,* 6.

Ebrahimi N.B. 1999. How to improve the calibration of cost models. *IEEE Transactions on Software Engineering,* 25(1), 136 - 140.

Engel A., Last M. 2007. Modeling software testing costs and risks using fuzzy logic paradigm. *Journal of Systems and Software,* 80(6), 817-835.

Eung S. J., Jae K. L. 2001. Quasi-optimal case-selective neural network model for software effort estimation. *Expert Systems with Applications*, 21(1), 1-14.

Fairley R. E. 2007. The influence of COCOMO on software engineering education and training. *Journal of Systems and Software*, 80(8), 1201-1208.

Finnie G. R., Wittig G. E., Desharnais J. M. 1997. A comparison of software effort estimation techniques: Using function points with neural networks, case-based reasoning and regression models. *Journal of Systems and Software,* 39(3), 281-289.

Fioravanti F., Nesi P. 2001 Estimation and prediction metrics for adaptive maintenance effort of object-oriented systems. *IEEE Transactions on Software Engineering*, 27(12), 1062-1084.

Foss T., Stensrud E., Kitchenham B., and Myrtveit I. 2003. A Simulation Study of the Model Evaluation Criterion MMRE. *IEEE Transactions on Software Engineering*, 29(11), 985-995.

Gelman A., Carlin J.B., Stern H.S., Rubin D.B. 1998. *Bayesian Data Analysis*. Chapman & Hall.

Gray A. R., MacDonell S. G. 1997. A comparison of techniques for developing predictive models of software metrics. *Information and Software Technology*, 39(6), 425-437.

Gray A. R., MacDonell S. G. 1999. Software metrics data analysis-exploring the relative performance of some commonly used modeling techniques. *Empirical Software Engineering*. 4(4), 297-316.

Grimstad S., Jørgensen M. 2007. Inconsistency of expert judgment-based estimates of software development effort. *Journal of Systems and Software*, 80(11), 1770-1777.

Hardy R. L. 1971 Multiquadratic equations of topography and other irregular surfaces. *Journal of Geophysics Research*. 76, 1905–1915.

Heiat A. 2002. Comparison of artificial neural network and regression models for estimating software development effort. *Information and Software Technology*, 44(15), 911-922.

Hsia P., Hsu C. T., Kung D.C., Byrne E.J. 1998 Incremental delivery reduces maintenance cost: a COCOMO-based study. *Journal of Software Maintenance: Research and Practice*, 10(4), 225-247.

Huang S. J., Chiu N, H. 2006. Optimization of analogy weights by genetic algorithm for software effort estimation. *Information and Software Technology* 48(11), 1034 – 1045.

Huang X. S., Ho D., Ren J., Capretz L. F. 2007. Improving the COCOMO model using a neuro-fuzzy approach. *Applied Soft Computing*, 7(1), 29–40.

Hughes R.T. 1996. Expert judgment as an estimating method. *Information and software technology*, 38(2), 67-75.

Hughes R.T., Cunliffe A., Young-Martos F. 1998. Evaluating software development effort model-building techniques for application in a real-time telecommunications environment. *IEE Proceedings Software*, 145(1), 29-33.

Idri A., Khoshgoftaar T. M., Abran A. 2002. Investigating soft computing in case-based reasoning for software cost estimation. *Engineering Intelligent Systems for Electrical Engineering and Communications*, 10(3), 147-157.

Jeffery R., Ruhe M., Wieczorek I., 2000. A comparative study of two software development cost modeling techniques using multi-organizational and company-specific data, *Information and Software Technology* 42(14), 1009－1016.

Jongmoon B., Boehm B., Steece B. M. 2002. Disaggregating and calibrating the CASE tool variable in COCOMO II. *IEEE Transactions on Software Engineering*, 28(11), 1009-1022.

Jorgensen M., 1995. An empirical study of software maintenance tasks. *Journal of Software Maintenance: Research and Practice*, 7(1), 27–48.

Jorgensen, M. 1995 Experience with the accuracy of software maintenance task effort prediction models. *IEEE Transactions on Software Engineering*, 21(8), 674-681.

Jorgensen M. 2004 a. A review of studies on expert estimation of software development effort. *Journal of Systems and Software,* 70(1-2), 37－60.

Jorgensen M. 2004 b. Realism in assessment of effort estimation uncertainty: It matters how you ask. *IEEE Transactions on Software Engineering*, 30(4), 209-217.

Jorgensen M. 2004 c. Regression models of software development effort estimation accuracy and bias. *Empirical Software Engineering*, 9(4), 297－314.

Jorgensen M. 2004 d. Top-down and bottom-up expert estimation of software development effort. *Information and Software Technology*, 46(1), 3–16.

Jorgensen M. 2005. Evidence-based guidelines for assessment of software development cost uncertainty. *IEEE Transactions on Software Engineering*, 31(11), 942 - 954.

Jorgensen M., Faugli B., Gruschke T. 2006. Characteristics of software engineers with optimistic predictions. *Journal of Systems and Software*, 80(9), 1472-1482.

Jorgensen M., Indahl U., Sjoberg D. 2003. Software effort estimation by analogy and 'regression toward the mean'. *Journal of Systems and Software*, 68(3), 253–262.

Jorgensen, M., Shepperd, M. 2007. A Systematic Review of Software Development Cost Estimation Studies. *IEEE Transactions on Software Engineering*. 33(1), 33 – 53.

Jorgensen M., Sjøberg D.I.K. 2003. An effort prediction interval approach based on the empirical distribution of previous estimation accuracy. *Information and Software Technology* 45(3), 123–136.

Jorgensen M., Sjoberg D. I. K. 2004. The impact of customer expectation on software development effort estimates. *International Journal of Project Management*, 22(4), 317－325.

Jorgensen M., Teigen K. H., Moloken K. 2004. Better sure than safe? Over-confidence in judgment based software development effort prediction intervals. *Journal of Systems and Software*, 70(1-2), 79－93.

Jorgensen M., Faugli B., Gruschke T. 2007. Characteristics of software engineers with optimistic predictions. *Journal of Systems and Software*, 80(9), 1472-1482.

Jun E. S., Lee J. K. 2001. Quasi-optimal case-selective neural network model for software effort estimation. *Expert Systems with Applications*, 21(1), 1-14.

Kadoda, G., Cartwright, M., Chen, L., Shepperd, M. 2000. Experiences using case-based reasoning to predict software project effort. *Proceedings EASE 2000 conferences – 4th International Conference on Empirical Assessment and Evaluation in Software Engineering*. Staffordshire, U.K.

Kaplan, H. T. 1991. The Ada COCOMO cost estimating model and VASTT development estimates vs. actuals. *Vitro Technical Journal*, 9(1), 48-60

Kemerer C. F. 1987. An empirical validation of software cost estimation models. *Communications of the ACM*, 30, 416-429.

Khoshgoftaar T.M., Allen E.B., Naik A., Jones W.D., Hudepohl J.P., 1999. Using classification trees for software quality models: lessons learned. *International Journal of Software Engineering and Knowledge Engineering*, 9(2), 217-231.

Kitchenham B.A. 1990. Software Development Cost Models. *Software Reliability Handbook*, Elsevier Applied Science, New York. 333-376.

Kitchenham B.A. 1998. A Procedure for Analyzing Unbalanced Datasets. *IEEE Transactions on Software Engineering*, 24(4), 278-301.

Kitchenham B.A., MacDonell S.G., Pickard L.M., Shepperd M.J. 2001. What Accuracy Statistics Really Measure," *IEE Proceeding Software Engineering*, 148(3), 81-85.

Kitchenham B.A., Linkman S. 1997. Estimates, Uncertainty and Risk, *IEEE Software*, 14(3), 69-74.

Kitchenham B.A., Pfleeger S. L., McColl B., Eagan S. 2002. An empirical study of maintenance and development estimation accuracy, *Journal of Systems and Software*, 64(1), 57-77.

Kwak, N., Choi C. H. 2002(a) Input feature selection for classification problems. *IEEE Transactions on Neural Networks*, 13(1), 143-159

Kwak, N., Choi, C. H. 2002(b) Input feature selection by mutual information based on parzen window. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12), 1667–1671.

Lawrence J. 1994. *Introduction to Neural Networks: Design, Theory, and Applications*. California Scientific Software. Nevada City, CA.

Leach R.J. 1996 Methods of measuring software reuse for the prediction of maintenance effort. *Journal of Software Maintenance: Research and Practice*, 8(5), 309-320.

Lee A., Chun H., Cheng, Balakrishnan J., 1998. Software development cost estimation: integrating neural network with cluster analysis. *Information and Management*, 34(1), 1-9.

Leung, H.K.N. 2002. Estimating maintenance effort by analogy. *Empirical Software Engineering*, 7(2), 157-75.

Li W., Henry S. 1993. Object-oriented metrics that predict maintainability. *Journal of Systems and Software* 23(2), 111–122.

Li J. Z., Ruhe G., Al-Emran A., Richter M. 2007. A flexible method for software effort estimation by analogy. *Empirical Software Engineering*, 12(1), 65-106.

Li J. Z., Ruhe G. 2008. Analysis of attribute weighting heuristics for analogy-based software effort estimation method AQUA+. *Empirical Software Engineering*, 13(1), 63-96.

Li J. Z., Ruhe G. 2008. Software effort estimation by analogy using attribute selection based on rough set analysis. *International Journal of Software Engineering and Knowledge Engineering*, 18(1), 1-23.

Li Y. F., Xie M., Goh T. N. 2007. A study of genetic algorithm for project selection for analogy based software cost estimation. *Proceedings of the IEEE International Conference on Industrial Engineering and Engineering Management 2007*, 1256-1260.

Li Y. F., Xie M., Goh T. N. 2008a. A Bayesian inference approach for probabilistic analogy based software maintenance effort estimation. *Proceedings of the Pacific Rim International Symposium on Dependable Computing 2008*, 176-183.

Li Y. F., Xie M., Goh T. N. 2008b. A study of analogy based sampling for interval based cost estimation for software project management. *Proceedings of the International Conference on Management of Innovation and Technology 2008*, 281-286.

Li Y. F., Xie M., Goh T. N. 2008c. Optimization of feature weights and number of neighbors for analogy based cost estimation in software project management. *Proceedings of the IEEE International Conference on Industrial Engineering and Engineering Management 2008*, 1542-1546.

Li Y. F., Xie M., Goh T. N. 2009a. A study of mutual information based feature selection for case based reasoning in software cost estimation. *Expert Systems with Applications*. 36(3), 5921-5931.

Li Y. F., Xie M., Goh T. N. 2009b. A study of project selection and feature weighting for analogy based software cost estimation. *Journal of Systems and Software*. 82(2), 241-252.

Li Y. F., Xie M., Goh T. N. 2009c. A study on the non-linear adjustment for analogy based software cost estimation. *Empirical Software Engineering*. 14(6), 1382-3256.

Liu Q. Mintram R. C. 2005. Preliminary data analysis methods in software estimation. *Software Quality Journal*, 13, 91–115.

Lo B.W.N., Gao X. Z. 1997. Assessing software cost estimation models: criteria for accuracy, consistency and regression. *Australian Journal of Information Systems*, 5(1), 30-44.

Lucia A., Pompella E., Stefanucci S. 2005. Assessing effort estimation models for corrective maintenance through empirical studies. *Information and Software Technology*, 47(1), 3-15.

MacDonell S.G. 1994. Comparative Review of Functional Complexity Assessment Methods for Effort Estimation. *Software Engineering Journal*, 9(3), 107-116.

Macdonell S.G., Fletcher T., Wong B.L.W. 1999. Industry practices in project management for multimedia information systems. *International Journal of Software Engineering and Knowledge Engineering*, 9(6), 801-815.

Macdonell S. G., Shepperd M. J. 2003. Combining techniques to optimize effort predictions in software project management. *Journal of Systems and Software*, 66(2), 91 - 98.

Mair C., Shepperd M., Jorgensen M. 2005. An analysis of data sets used to train and validate cost prediction systems. *Proceedings of the 2005 workshop on Predictor Models in Software Engineering*, 1 – 6.

Mair C., Kadoda G., Lefley M., Phalp K., Schofield C., Shepperd M., Webster S. 2000.

An investigation of machine learning based prediction systems, *Journal of Systems and Software*, 53(1), 23-29.

Marbán O., Menasalvas E., Fernández-Baizán C. 2007. A cost model to estimate the effort of data mining projects (DMCoMo). *Information Systems*, 33(1), 133 – 150.

Matson, J.E., Barrett, B.E., and Mellichamp, J.M. 1994. Software development cost estimation using function points, *IEEE Transactions on Software Engineering*, 20(4), 275–287.

McDonald J. 2005. The impact of project planning team experience on software project cost estimates. *Empirical Software Engineering*, 10(2), 219-234.

Mendes E., Watson I., Triggs C., Mosley N., Counsell S. 2003. A comparative study of cost estimation models for Web hypermedia applications. *Empirical Software Engineering*, 8, 163 - 196.

Mendes E., Mosley N., Counsell S. 2005. Investigating Web size metrics for early Web cost estimation. *Journal of Systems and Software*, 77(2), 157-72.

Mendes, E., Di Martino, S., Ferrucci, F., Gravino, C. 2008. Cross-company vs. single-company Web effort models using the Tukutuku database: an extended study. *Journal of Systems and Software*, 81(5), 673-690.

Mendes, E., Lokan, C. 2008. Replicating studies on cross- vs single-company effort models using the ISBSG Database. *Empirical Software Engineering*, 13(1), 3-37.

Menzies T., Chen Z., Hihn J., Lum K. 2008. Selecting best practices for effort estimation. *IEEE Transactions on Software Engineering*, 32(11), 883-895.

Mittas, N., Athanasiades, M., Angelis, L. 2008 Improving analogy-based software cost estimation by a resampling method. *Information and Software Technology*, 50(3), 221-230.

Miyazaki Y., Terakado K., Ozaki K., Nozaki H. 1994. Robust regression for developing software estimation models. *Journal of Systems and Software*, 27, 3-16.

Moløkken-Ostvold K., Jørgensen M. 2004. Group Processes in Software Effort Estimation. *Empirical Software Engineering*, 9(4), 315-334.

Moløkken K., Jørgensen M. 2005. Expert Estimation of Web-Development Projects: Are Software Professionals in Technical Roles More Optimistic Than Those in Non-Technical Roles? *Empirical Software Engineering*, 10(1), 7-30.

Moses, J. 2002. Measuring Effort Estimation Uncertainty to Improve Client Confidence. *Software Quality Journal*, 10, 135-148.

Moses J., Farrow M. 2003. A Procedure for Assessing the Influence of Problem Domain on Effort Estimation Consistency. *Software Quality Journal*, 11(4), 283-300.

Moses J., Farrow M. 2005. Assessing Variation in Development Effort Consistency Using a Data Source with Missing Data. *Software Quality Journal*, 13(1), 71-89.

Musilek P., Pedrycs W., Succi G., Reformat M. 2000. Software cost estimation with fuzzy models. *Applied Computing Review*, 8(2), 24-29.

Myrtveit Y., Stensrud E., Shepperd M. 2005. Reliability and Validity in Comparative Studies of Software Prediction Models. *IEEE Transactions on Software Engineering* 31(5) 380-391

NASA, 1990. Manager's Handbook for Software Development, Goddard Space Flight Center, NASA Software Engineering Laboratory, Greenbelt, MD.

Oliveira A. L.I. 2006. Estimation of software project effort with support vector regression.

Neurocomputing, 69(13-15),1749 – 1753.

Park B. J., Pedrycz W., Oh S. K. 2008. An approach to fuzzy granule-based hierarchical polynomial networks for empirical data modeling in software engineering. *Information and Software Technology*, 50(9-10), 912-923.

Pearl J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Pendharkar P. C., Subramanian G. H. 2002. Connectionist models for learning, discovering, and forecasting software effort: an empirical study. *Journal of Computer Information Systems*, 43(1), 7-14

Pendharkar P. C., Subramanian G. H., Rodger J. A. 2005. A Probabilistic Model for Predicting Software Development Effort. *IEEE Transactions on Software Engineering*, 31(7), 615-624.

Pendharkar P. C., Subramanian G. H. 2006. An empirical study of ICASE learning curves and probability bounds for software development effort. *European Journal of Operational Research*. 183(3), 1086 – 1096.

Peng H. C. 2007. http://www.mathworks.com/matlabcentral/fileexchange/14888

Pengelly A., 1995. Performance of effort estimating techniques in current development environments. *Software Engineering Journal*, 10 (5), 162–170.

Pickard L., Kitchenham B., Linkman S., 1999. An investigation analysis techniques for software datasets, *Proceedings of 6th International Software Metrics Symposium*, 130 – 142.

Pickard L., Kitchenham B., Linkman S. 2001. Using simulated data sets to compare data analysis techniques used for software cost modeling. *IEE Proceedings of Software* 148(6).

Putnam L., Myers W. 1992. *Measures for Excellence*, Yourdon Press Computing Series.

Ramanujan S., Scamell R. W., Shah J. R. 2000 An experimental investigation of the impact of individual, program, and organizational characteristics on software maintenance effort. *Journal of Systems and Software*, 54(2), 137-157.

Rousseeuw P. J., Leroy A. M. 1987. *Robust Regression and Outlier Detection*, Wiley, New York.

Samson B., Ellison D., Dugard P. 1997. Software cost estimation using an Albus perceptron (CMAC). *Information and Software Technology*, 39(1), 55-60.

Schank R. C. 1982. *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*. Cambridge University Press, Cambridge.

Schroeder L., Sjoquist D., Stephan, P. 1986. Understanding Regression Analysis: An Introductory Guide. No. 57. In Series: *Quantitative Applications in the Social Sciences*, CA, USA: Sage Publications, Newbury Park.

Selby R., Boehm B. 2007. *Software engineering: Barry W. Boehm's lifetime contributions to software development, management, and research*. Wiley-IEEE Computer Society.

Sentas P., Angelis L., Stamelos I., Bleris G. 2005. Software productivity and effort prediction with ordinal regression. *Information and Software Technology*, 47(1), 17–29.

Sentas P., Angelis L. 2006. Categorical missing data imputation for software cost estimation by multinomial logistic regression. *Journal of Systems and Software*, 79(3), 404-414.

Shepperd M., Schofield C. 1997. Estimating Software Project Effort Using Analogies. *IEEE Transactions on Software Engineering*, 23(12), 733-743.

Shepperd M., Cartwright M., Kadoda G. 2000. On building prediction systems for software engineers. *Empirical Software Engineering*. 5(3), 175-182

Shepperd M., Kadoda G. 2001. Comparing software prediction techniques using simulation. *IEEE Transactions on Software Engineering*, 27(11), 1014-1022.

Shin M., Goel A.L. 2000. Empirical Data Modeling in Software Engineering Using Radial Basis Functions. *IEEE Transactions on Software Engineering,* 26(6), 567 – 576.

Shukla K. K. 2000. Neuro-genetic prediction of software development effort. *Information and Software Technology*, 42(10), 701-713.

Song Q. B., Shepperd M., Cartwright M. 2005. A short note on safest default missingness mechanism assumptions. *Empirical Software Engineering,* 10(2), 235-243.

Song Q. B., Shepperd M. 2007. A new imputation method for small software project data sets. *Journal of Systems and Software*, 80, 51–62.

Srinivasan R., Fisher D. 1995. Machine Learning Approaches to Estimating Software Development Effort, *IEEE Transactions on Software Engineering*. 21(2), 126-137.

Stamelos I., Angelis L. 2001. Managing uncertainty in project portfolio cost estimation. *Information and Software Technology*, 43(13), 759-768.

Stamelos I., Angelis L., Dimou P., Sakellaris E. 2003. On the use of Bayesian belief networks for the prediction of software productivity. *Information and Software Technology*, 45(1), 51-60.

Standing G. 2004. *CHAOS*, 2004, http://www.projectsmart.co.uk/docs/chaos_report.pdf.

Stensrud, E.2001. Alternative approaches to effort prediction of ERP projects. *Information and Software Technology*, 43(7), 413-423.

Stensrud E., Foss T., Kitchenham B., Myrtveit I. 2003. A further empirical investigation of the relationship between MRE and project size. *Empirical Software Engineering*. 8(2), 139-161.

Sternberg, R., 1977. Component processes in analogical reasoning. *Psychological Review* 84 (4), 353 – 378.

Stewart B. 2002. Predicting project delivery rates using the Naive-Bayes classifier. *Journal of Software Maintenance and Evolution: Research and Practice*, 14(3), 161-179.

Strike K., El-Emam K., Madhavji N. 2001. Software cost estimation with incomplete data. *IEEE Transactions on Software Engineering,* 27(10), 890-908.

Van Koten C., Gray A.R. 2006. Bayesian statistical effort prediction models for data-centred 4GL software development, *Information and Software Technology*, 48(11), 1056-1067.

Van Koten C., Gray A.R. 2006 An application of Bayesian network for predicting object-oriented software maintainability. *Information and Software Technology*, 48(1), 59-67.

Vapnik V. 1995. *The nature of statistical learning theory*. New York: Springer.

Walkerden F., Jeffery D. R. 1999. An empirical study of analogy-based software effort estimation. *Empirical Software Engineering*, 4, 135–158.

Xu Z., Khoshgoftaar T. M. 2004. Identification of fuzzy models of software cost estimation. *Fuzzy Sets and Systems*, 145(1), 141-163.

Yu L.G. 2006 Indirectly predicting the maintenance effort of open-source software. *Journal of Software Maintenance and Evolution*, 18(5), 311-332.

Zhou Y. M., Leung H. 2007 Predicting object-oriented software maintainability using multivariate adaptive regression splines. *Journal of Systems and Software*, 80(8), 1349-1361.

Appendix A

Table A.1: Journal publications under each method 1999-2008

Year	EJ	PM	RE	ML	AB	Others
1999	(Myrtveit and Stensrud, 1999, Chulani et al., 1999)	(Moser et al., 1999) (Ferens, 1999, Chulani et al., 1999)	(Ebrahimi, 1999, Myrtveit and Stensrud, 1999, Moser et al., 1999, Maxwell et al., 1999)	(Chulani et al., 1999)	(Myrtveit and Stensrud, 1999)	(Padberg, 1999, Schooff and Haimes, 1999)
2000	(Boehm et al., 2000)	(Boehm et al., 2000, Lokan, 2000)	(Boehm et al., 2000, Jeffery et al., 2000, Costagliola et al., 2000, Dolado, 2000, Fioravanti and Nesi, 2000)	(Boehm et al., 2000, Mair et al., 2000, Shin and Goel, 2000, Dolado, 2000, Shukla, 2000)	(Boehm et al., 2000, Jeffery et al., 2000, Mair et al., 2000)	(Lederer and Prasad, 2000, Runeson et al., 2000)
2001	(Jorgensen and Sjoberg, 2001, Shepperd and Cartwright, 2001, Miranda, 2001)	(Hastings and Sajeev, 2001, Caban et al., 2001, Smith et al., 2001, Jun and Lee, 2001)	(Fioravanti and Nesi, 2001, Burgess and Lefley, 2001, Moores, 2001, Briand and Wust, 2001, Hastings and Sajeev, 2001, Caban et al., 2001, Dolado, 2001, Myrtveit et al., 2001)	(Kadoda et al., 2001, Burgess and Lefley, 2001, Mizuno et al., 2001, Briand and Wust, 2001, Jun and Lee, 2001, Dolado, 2001)	(Kadoda et al., 2001, Burgess and Lefley, 2001, Stamelos and Angelis, 2001, Jun and Lee, 2001, Strike et al., 2001)	(Myrtveit et al., 2001, Strike et al., 2001, Calzolari et al., 2001)
2002		(Smith, 2002, Kitchenham et al., 2002, Abran et al., 2002, Baik et al., 2002)	(Kitchenham et al., 2002, Heiat, 2002, Abran et al., 2002)	(Heiat, 2002, Pendharkar and Subramanian, 2002, Moses, 2002, Oh et al., 2002)	(Idri et al., 2002)	(Koch and Schneider, 2002)

Year	EJ	PM	RE	ML	AB	Others
2003	(Jorgensen et al., 2003, Jorgensen and Sjoberg, 2003, MacDonell and Shepperd, 2003, Stamelos et al., 2003a)	(Ahn et al., 2003, Benediktsson et al., 2003)	(Mendes et al., 2003, Ahn et al., 2003, De Lucia et al., 2003, MacDonell, 2003, MacDonell and Shepperd, 2003, Moses and Farrow, 2003)	(Mendes et al., 2003, Lefley and Shepperd, 2003, MacDonell, 2003, Moses and Farrow, 2003, Stamelos et al., 2003a)	(Mendes et al., 2003, Jorgensen et al., 2003, Jorgensen and Sjoberg, 2003, Kirsopp et al., 2003, MacDonell and Shepperd, 2003, Stamelos et al., 2003b)	(Stensrud et al., 2003, Staub-French et al., 2003)
2004	(Jorgensen, 2004d, Jorgensen, 2004c, Jorgensen, 2004a, Jorgensen and Molokken, 2004, Jorgensen and Molokken-Ostvold, 2004, Jorgensen et al., 2004, Molokken-Ostvold and Jorgensen, 2004)	(Xu and Khoshgoftaar, 2004)	(Jorgensen, 2004b, Kaczmarek and Kucharski, 2004, Kitchenham and Mendes, 2004)	(Oh et al., 2004b, Oh et al., 2004a, Xu and Khoshgoftaar, 2004)	(Ohsugi et al., 2004)	(Benediktsson and Dalcher, 2004, Molokken et al., 2004)
2005	(Jorgensen, 2005b, Jorgensen, 2005a, Molokken-Ostvold and Jorgensen, 2005)	(Ahmed et al., 2005, Ben Lamine et al., 2005, De Lucia et al., 2005)	(Myrtveit et al., 2005, Sentas et al., 2005, Mendes et al., 2005, Liu and Mintram, 2005, McDonald, 2005, Moses and Farrow, 2005)	(Ahmed et al., 2005, MacDonell and Gray, 2005, Moses and Farrow, 2005, Pendharkar et al., 2005, Sicilia et al., 2005, Musilek and Meltzer, 2005)	(Myrtveit et al., 2005)	

Year	EJ	PM	RE	ML	AB	Others
2006	(Cuadrado-Gallego et al., 2006, Jorgensen and Molokken-Ostvold, 2006)	(Cuadrado-Gallego et al., 2006a, Wehrmann and Gull, 2006, Subramanian et al., 2006, Costagliola et al., 2006, Cuadrado-Gallego et al., 2006b, Choi and Sircar, 2006)	(Huang and Chiu, 2006, Iwata et al., 2006, Oliveira, 2006, Yu, 2006, Sentas and Angelis, 2006)	(Crespo and Marban, 2006, Huang and Chiu, 2006, Huang et al., 2006a, Huang et al., 2006b, Oliveira, 2006, Song et al., 2006, Stefanowski, 2006, van Koten and Gray, 2006)	(Auer et al., 2006, Huang and Chiu, 2006, Lee and Lee, 2006)	(Grimstad et al., 2006, Issa et al., 2006, Rodriguez et al., 2006, Menzies and Hihn, 2006)
2007	(Jorgensen and Shepperd, 2007, Kitchenham et al., 2007, Grinistad and Jorgensen, 2007)	(Jorgensen and Shepperd, 2007, Kitchenham et al., 2007, Fairley, 2007, Huang et al., 2007, Gallego et al., 2007, Cuadrado-Gallego and Sicilia, 2007)	(Chiu and Huang, 2007) (Jorgensen and Shepperd, 2007, Kitchenham et al., 2007, Morgenshtern et al., 2007, Cuadrado-Gallego and Sicilia, 2007, Bourque et al., 2007, Baresi and Morasca, 2007, Agrawal and Chari, 2007)	(Chiu and Huang, 2007, Huang et al., 2007, Jorgensen and Shepperd, 2007, Kitchenham et al., 2007, Gallego et al., 2007)	(Chiu and Huang, 2007, Li et al., 2007, Jorgensen and Shepperd, 2007, Kitchenham et al., 2007, Song and Shepperd, 2007)	(Song and Shepperd, 2007, Pendharkar and Subramanian, 2007, Kouskouras and Georgiou, 2007)
2008	(Park and Baek, 2008, Gruschke and Jorgensen, 2008, Boehm and Valerdi, 2008)	(Tronto et al., 2008, Marban et al., 2008, Boehm and Valerdi, 2008)	(Tronto et al., 2008, Park and Baek, 2008, Park et al., 2008, Mittas and Angelis, 2008, Mendes and Mosley, 2008, Mendes et al., 2008, Lopez-Martin et al., 2008, Kumar et al., 2008, Huang et al., 2008b, Capra et al., 2008)	(Tronto et al., 2008, Park and Baek, 2008, Park et al., 2008, Moreno Garcia et al., 2008, Mendes and Mosley, 2008, Lopez-Martin et al., 2008, Kumar et al., 2008, Huang et al., 2008a, Bibi et al., 2008, Aroba et al., 2008)	(Song et al., 2008, Mittas et al., 2008, Mittas and Angelis, 2008, Mendes and Mosley, 2008, Mendes et al., 2008, Li and Ruhe, 2008b, Li and Ruhe, 2008a, Keung et al., 2008, Huang et al., 2008a)	(Xia et al., 2008, Daneva and Wieringa, 2008)

Appendix B

Table B.1: Feature definition of Albrecht dataset

Features	Full name	Type	Description
Inpcount	Input count	*Numerical*	Count of inputs
Outcount	Output count	*Numerical*	Count of outputs
Quecount	Query count	*Numerical*	Count of queries
Filcount	File count	*Numerical*	Count of files
Fp	Function points	*Numerical*	Number of function points
SLOC	Lines of source code	*Numerical*	Lines of source code
Effort	Development effort	*Numerical*	Measured in 1000 hours

Table B.2: Descriptive statistics of all features of Albrecht dataset

Features	Mean	Std Dev	Min	Max	Skewness	Kurtosis
Inpcount	40.25	36.91	7.00	193.00	3.07	13.44
Outcount	47.25	35.17	12.00	150.00	1.28	4.29
Quecount	17.38	15.52	3.00	60.00	1.40	3.96
Filcount	16.88	19.34	0	75.00	1.94	6.46
Fp	61.08	63.68	3.00	318.00	2.90	12.19
SLOC	199.00	1902.00	647.63	488.00	1.44	4.02
Effort	21.88	28.42	0.50	105.20	2.16	6.51

Table B.3: Feature definition of Desharnais dataset

Features	Full name	Type	Description
TeamExp	Team experience	*Numerical*	Measured in years
ManagerExp	Manager's experience	*Numerical*	Measured in years
YearEnd	Year of end	*Numerical*	The ending year of development
Length	Length of project	*Numerical*	The number of years used for development
Transactions	Transactions	*Numerical*	Number of transactions
Entities	Entities	*Numerical*	Number of entities
PointsNonAdjust	Non-adjusted function points	*Numerical*	Number of non-adjusted function points
PointsAdjust	Adjusted function points	*Numerical*	Number of adjusted function points
Envergure	Development environment	*Numerical*	Development environment
Language	Programming language	*Categorical*	1 = 1st generation 2 = 2nd generation 3 = 3rd generation
Effort	Development effort	*Numerical*	Measured in 1000 hours

Table B.4 Descriptive statistics of all features of Desharnais dataset

Features	Mean	Std Dev	Min	Max	Skewness	Kurtosis
TeamExp	2.30	1.33	0	4.00	-0.05	1.73
ManagerExp	2.65	1.52	0	7.00	0.22	3.01
YearEnd	85.78	1.14	83.00	88.00	-0.20	3.05
Length	11.30	6.79	1.00	36.00	1.43	5.49
Language	1.56	0.72	1.00	3.00	0.88	2.45
Transactions	177.47	146.08	9.00	886.00	2.34	10.09
Entities	120.55	86.11	7.00	387.00	1.36	4.37
Envergure	27.45	10.53	5.00	52.00	-0.19	2.58
PointsNonAdjust	282.39	186.36	62.00	1116.00	1.70	7.08
PointsAdjust	298.01	182.26	73.00	1127.00	1.81	7.67
Effort	4.83	4.189	0.55	23.94	2.00	7.89

Table B.5 Feature definition in Maxwell dataset

Features	Full name	Type	Description
Time	Time	*Numerical*	Time = syear – 1985 + 1, with levels: 1,2,3,4,5,6,7,8,9..
App	Application type	*Categorical*	1 = Information/on-line service (infServ)
			2 = Transaction control, logistics, order processing (TransPro)
			3 = Customer service (CustServ)
			4 = Production control, logistics, order processing (ProdCont)
			5 = Management information system (MIS)
Har	Hardware platform	*Categorical*	1 = Personal computer (PC)
			2 = Mainframe (Mainfrm)
			3 = Multi-platform (Multi)
			4 = Mini computer (Mini)
			5 = Networked (Network)
Dba	Database	*Categorical*	1 = Relatnl (Relational)
			2 = Sequentl (Sequential)
			3 = Other (Other)
			4 = None (None)
Ifc	User interface	*Categorical*	1 = Graphical user interface (GUI)
			2 = Text user interface (TextUI)
Source	Where developed	*Categorical*	1 = In-house (Inhouse)
			2 = Outsourced (Outsrced)
Telonuse	Telon use	*Categorical*	0 = No
			1 = Yes
Nlan	Number of different development languages used	*Ordinal*	1 = one language used
			2 = two languages used
			3 = three languages used
			4 = four languages used
T01	Customer participation	*Ordinal:*	1 = Very low
T02	Development environment adequacy		2 = Low
T03	Staff availability		3 = Nominal
T04	Standards use		4 = High
T05	Methods use		5 = Very high
T06	Tools use		
T07	Software's logical complexity		
T08	Requirements volatility		

Features	Full name	Type	Description
T09	Quality requirements		
T10	Efficiency requirements		
T11	Installation requirements		
T12	Staff analysis skills		
T13	Staff application knowledge		
T14	Staff tool skills		
T15	Staff team skills		
Duration	Duration	*Numerical*	Duration of project from specification until delivery, measured in months
Size	Application size	*Numerical*	Function points measured using the experience method
Effort	Effort	*Numerical*	Work carried out by the software supplier from specification until delivery, measured in hours

Table B.6 Descriptive statistics of all features of Maxwell data set

Features	Mean	Std Dev	Min	Max	Skewness	Kurtosis
Time	5.58	2.13	1.00	9.00	-0.42	2.25
App	2.35	0.99	1.00	5.00	0.96	4.11
Har	2.61	1.00	1.00	5.00	1.43	4.09
Dba	1.03	0.44	0.00	4.00	4.74	35.13
Ifc	1.94	0.25	1.00	2.00	-3.55	13.57
Source	1.87	0.34	1.00	2.00	-2.21	5.90
Telonuse	2.55	1.02	1.00	4.00	-0.04	1.91
Nlan	0.24	0.43	0.00	1.00	1.21	2.45
T01	3.05	1.00	1.00	5.00	-0.20	2.05
T02	3.05	0.71	1.00	5.00	-0.07	3.57
T03	3.03	0.89	2.00	5.00	0.51	2.51
T04	3.19	0.70	2.00	5.00	0.02	2.60
T05	3.05	0.71	1.00	5.00	0.48	4.98
T06	2.90	0.69	1.00	4.00	-0.46	3.49
T07	3.24	0.90	1.00	5.00	-0.08	2.52
T08	3.81	0.96	2.00	5.00	-0.17	1.97
T09	4.06	0.74	2.00	5.00	-0.58	3.32
T10	3.61	0.89	2.00	5.00	0.00	2.22
T11	3.42	0.98	2.00	5.00	0.12	2.02
T12	3.82	0.69	2.00	5.00	-0.66	3.83
T13	3.06	0.96	1.00	5.00	-0.24	2.35
T14	3.26	1.01	1.00	5.00	-0.15	2.37
T15	3.34	0.75	1.00	5.00	0.09	3.99
Duration	17.21	10.65	4.00	54.00	1.25	4.34
Size	673.31	784.08	48.00	3643.00	2.28	7.80
Effort	8223.21	10499.90	583.00	63694.00	3.27	15.52

Table B.7 Feature definition in ISBSG dataset

Features	Full name	Type	Description
DevType	Development type	*Categorical*	1 = Enhancement 2 = New development 3 = Re-development

OrgType	Organization type	*Categorical*	1 = Banking
			2 = Communication
			3 = Community services
			4 = Computer, Software, ISP
			5 = Electricity, Gas, Water;
			6 = Financial, Property & Business Services;
			7 = Insurance;
			8 = Manufacturing;
			9 = Government, Public Administration
			10 = Transport & Storage;
			11 = Wholesale & Retail Trade;
			12 = Others.

BusType	Business Area Type	*Categorical*	1 = Accounting;
			2 = Banking;
			3 = Engineering;
			4 = Financial;
			5 = Insurance, Actuarial;
			6 = Inventory;
			7 = Legal;
			8 = Logistics;
			9 = Manufacturing
			10 = Personnel;
			11 = Research & Development;
			12 = Sales & Marketing;
			13 = Telecommunications;
			14 = Others.

AppType	Application Type	*Categorical*	1 = Billing;
			2 = Office information system, Executive information system, Decision support system;
			3 = Electronic Data Interchange;
			4 = Financial;
			5 = Management Information System;
			6 = Network Management, Communications;
			7 = Process control, sensor control, real time;
			8 = Transaction/Production System;
			9 = Others.

DevPlat	Development Platform	*Categorical*	1 = Mainframe 2 = Mid-range

			3 = Multi; 4 = Personal Computer.
Features	Full name	Type	Description

PriProLan	Primary Programming Language	*Categorical*	1 = ABAP;
			2 = Access;
			3 = ASP;
			4 = C;
			5 = C++;
			6 = COBOL;
			7 = JAVA;
			8 = Lotus Notes;
			9 = NATURAL;
			10 = ORACLE;
			11 = PL/I;
			12 = PL/SQL;
			13 = PowerBuilder;
			14 = SQL;
			15 = Visual Basic;
			16 = Others.

DevTech	Development Techniques	*Categorical*	1 = Business area modeling;
			2 = Data Modelling;
			3 = Event Modelling
			4 = Joint Application Development;
			5 = Multifunction teams
			6 = Object Oriented Analysis;
			7 = Object Oriented Design;
			8 = Process Modelling;
			9 = Prototyping;
			10 = Rapid Application Development;
			11 = WaterFall;
			12 = Others.

InpCont	Input Count	*Numerical*	The count of inputs
OutCont	Output Count	*Numerical:*	The count of outputs
EnqCont	Enquiry count	*Numerical:*	The count of enquiries
FileCont	File count	*Numerical:*	The count of files
IntCont	Interface count	*Numerical:*	The count of interfaces
AFP	Adjusted function points	*Numerical:*	The adjusted function point-count number
NorEffort	Normalized Work Effort	*Numerical:*	For project covering less than a full development life cycle, this value is an estimate of the full development effort in hours.

Table B.8 Descriptive statistics of all features of ISBSG data set

Features	Mean	Std Dev	Min	Max	Skewness	Kurtosis
DevType	1.52	0.50	1.00	2.00	-0.07	1.00
BusType	7.55	6.36	2.00	15.00	0.29	1.11
AppType	5.76	2.14	1.00	9.00	0.18	1.85
DevPlat	6.25	4.50	1.00	12.00	0.03	1.12
PriProLan	1.45	0.77	1.00	4.00	1.87	6.07
DevTech	10.19	3.96	4.00	16.00	0.10	1.66
InpCont	75.05	128.38	0	780.00	3.37	15.78
OutCont	68.90	96.81	0	648.00	3.42	17.50
EnqCont	41.49	75.80	0	398.00	2.70	10.23
FileCont	61.25	79.03	0	383.00	2.24	8.23
IntCont	28.07	36.74	0	172.00	1.83	6.02
AFP	284.41	340.65	10.00	2190.00	2.81	12.63
NorEffort	4309.08	5520.68	508.00	36046.00	2.86	13.29

Table B.9 Definition of software metrics

Metric	Definition
DIT (Depth of inheritance tree)	The length of the longest path from a given class to the root in the inheritance hierarchy
NOC (Number of children)	The number of classes that directly inherit from a given class
MPC (Message-passing coupling)	The number of send statements defined in a given class
RFC (Response for a class)	The number of methods that can potentially be executed in response to a message being received by an object of a given class
LCOM (Lack of cohesion in methods)	The number of pairs of local methods in a given class using no attribute in common
DAC (Data abstraction coupling)	The number of abstract data types defined in a given class
WMC (Weighted methods per class)	The sum of McCabe's cyclomatic complexity of all local methods in a given class
NOM (Number of methods)	The number of methods implemented within a given class
SIZE1 (Lines of code)	The number of semicolons in a given class
SIZE2 (Number of properties)	The total number of attributes and the number of local methods in a given class
CHANGE (Number of lines changed in the class)	Insertion and deletion are independently counted as 1, change of the contents is counted as 2

Table B.10 Descriptive statistics of UIMS dataset

Metric	Maximum	Median	Minimum	Mean	Standard deviation	Skewness	Kurtosis
DIT	4	2	0	2.15	0.90	-0.54	0.09
NOC	8	0	0	0.95	2.01	2.24	4.28
MPC	12	3	1	4.33	3.41	0.731	-0.70
RFC	101	17	2	23.21	20.19	2.00	4.94
LCOM	31	6	1	7.49	6.11	2.49	6.86
DAC	21	1	0	2.41	4.00	3.33	12.87
WMC	69	5	0	11.38	15.90	2.03	3.98
NOM	40	7	1	11.38	10.21	1.67	1.94
SIZE1	439	74	4	106.44	114.65	1.71	2.04
SIZE2	61	9	1	13.97	13.47	1.89	3.44
CHANGE	289	18	2	46.82	71.89	2.29	4.35

Table B.11 Descriptive statistics of QUES dataset

Metric	Maximum	Median	Minimum	Mean	Standard deviation	Skewness	Kurtosis
DIT	4	2	0	1.92	0.53	-0.10	5.46
NOC	0	NA	0	0	0.00	NA	NA
MPC	42	17	2	17.75	8.33	0.88	1.17
RFC	156	40	17	54.44	32.62	1.62	1.96
LCOM	33	5	3	9.18	7.34	1.35	1.10
DAC	25	2	0	3.44	3.91	2.99	12.82
WMC	83	9	1	14.96	17.06	1.77	3.33
NOM	57	6	4	13.41	12.00	1.39	1.40
SIZE1	1009	211	115	275.58	171.60	2.11	5.23
SIZE2	82	10	4	18.03	15.21	1.71	3.42
CHANGE	217	52	6	64.23	43.13	1.36	2.17