Multiple Costs and Their Combination in Cost Sensitive Learning

By

Zhenxing Qin

Submitted in fulfilment of the requirement for the degree of

Doctor of Philosophy

University of Technology, Sydney

June 2006

CERTIFICATE OF AUTHORSHIP/ORIGINALITY

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Signature of Candidate
Production Note:
Signature removed prior to publication.

To my wife Yanfang and Our Parents

Abstract

Cost sensitive learning is firstly defined as a procedure of minimizing the costs of classification errors. It has attracted much attention in the last few years. Being cost sensitive has the strength to handle the unbalance on the misclassification errors in some real world applications. Recently, researchers have considered how to deal with two or more costs in a model, such as involving both of the *misclassification costs* (the cost for misclassification errors) and *attribute test costs* (the cost incurs as obtaining the attribute's value) [Tur95, GGR02, LYWZ04]. Cost sensitive learning involving both attribute test costs and misclassification costs is called ***test cost sensitive learning*** that is more close to real industry focus, such as medical research and business decision.

Current test cost sensitive learning aims to find an optimal diagnostic policy (simply, a policy) with minimal expected sum of the misclassification cost and test cost that specifies, for example which attribute test is performed in next step based on the outcomes of previous attribute tests, and when the algorithm stops (by choosing to classify). A diagnostic policy takes the form of a decision tree whose nodes specify tests and whose leaves specify classification actions. A challenging issue is the choice of a reasonable one from all possible policies.

This dissertation argues for considering both of the test cost and misclassification cost, or even more costs together, but doubts if the current way, summing up the two costs, is the only right way. Detailed studies are needed to ensure the ways of combination make sense and be "correct", dimensionally as well as semantically. This dissertation studies fundamental properties of costs involved and designs new models to combine the costs together.

Some essential properties of attribute test cost are studied. In our learning problem definition, test cost is combined into misclassification cost by choosing and performing proper tests for a better decision. Why do you choose them and how about the ones that are not chosen? Very often, only part of all attribute values are enough for making a decision and rest attributes are left as "unknown". The values are defined as '*absent values*' as they are left as unknown purposely for some rational reasons when the information obtained is considered as enough, or when patients have no money enough to perform further tests, and so on.. This is the first work to utilize the information hidden in those "absent values" in cost sensitive learning; and the conclusion is very positive, i.e. "Absent data" is useful for decision making. The "absent values" are usually treated as '*missing values*' when left as known for unexpected reasons. This thesis studies the difference between 'absent' and 'missing'. An algorithm based on lazy decision tree is proposed to identify the absent data from missing data, and a novel strategy is proposed to help patch the "real" missing values. .

Two novel test cost sensitive models are designed for different real work scenarios. The first model is a *general test cost sensitive learning framework with multiple cost scales*. Previous works assume that the test cost and the misclassification cost must be defined on the same cost scale, such as the dollar cost incurred in a medical diagnosis. And they aim to minimize the sum of the misclassification cost and the test cost. However, costs may be measured in very different units and we may meet difficulty in defining the multiple costs on the same cost scale. It is not only a technology issue, but also a social issue. In medical diagnosis, how much money should you assign for a misclassification cost? Sometimes, a misclassification may hurt a patient's life. And from a social point of view, life is invaluable. To tackle this issue, a *target-resource budget learning framework* with multiple costs is proposed. With this framework, we present a test cost sensitive decision tree model with two kinds of cost scales. The task is to minimize one cost scale, called target cost, and keep the other one within specified budgets. To the best of our knowledge, this is the first attempt to study the cost sensitive learning with multiple costs scales.

The second model is based on the assumption that some attributes of an unlabeled example are known before being classified. A test cost sensitive lazy tree model is proposed to utilize the known information to reduce the overall cost. We also modify and apply this model to the batch-test problem: multiple tests are chosen and done in one shot, rather than in a sequential manner in the test-sensitive tree. It is significant in some diagnosis applications that require a decision to be made as soon as possible, such as emergency treatment.

Extensive experiments are conducted for evaluating the proposed approaches, and demonstrate that the work in this dissertation is efficient and useful for many diagnostic tasks involving target cost minimization and resource utilization for obtaining missing information.

Acknowledgements

First, I would like to take this opportunity to express my sincere gratitude to my supervisor, Professor Chengqi Zhang, for his unreserved encouragement, advice and support, and for giving me the opportunity to pursue my PhD at the Faculty of Information Technology at the University of Technology, Sydney. In particular, when I did not meet the enrolment deadline in 2003 due to the IELTS and Visa delay, he made much effort to hold my scholarship for more than half a year until I had the honour of studying and working with him in the past three years, and that is stamped indelibly on my life. His comments are so helpful, and his suggestions are so challenging, that I profit a lot from his guidance during my PhD program. I am honoured and feel happy to be able to work under such a supervisor like Chengqi. His profound knowledge and wisdom have deeply impressed me. I will remember his kindness forever.

Also, I am full of gratitude to Dr. Shichao Zhang, my co-supervisor, for his great help on my daily life and study, and for his detailed and constructive advice on my researches. He is also my master supervisor in China and he recommends me to Professor Chengqi Zhang as his Ph. D student, i.e. he is always a pilot of my life road and guides my onward direction. I would not be able to complete this thesis without his selfless help. He has always provided me with his knowledgeable views.

I am grateful to the Faculty of Information Technology, the University of Technology, Sydney, for providing me with a nice opportunity, an excellent environment and scholarship for the learning and researching here. I am grateful to Dr. Jie Lu and Dr. Guangquan Zhang for their kindl support and valuable discussion.

My thanks also go to all those who have helped me in one way or another during my PhD course: Ms. Li Liu, Ms Yanchun Zhou, Mr. Xiaowei Yan, Mr. Longbing Cao, Mr. Qingfeng Cheng, Mr. Chunsheng Li, Mr. Li Lin, Mr. Jiaqi Wang, Mr. Wanli Chen, Mr. Jiarui Ni, Mr. Yanchang Zhao, Mr. Jiarui Ni, Mr. Chengeng Shi, Mr. Xuetao Guo, and Mr. Alan Tao Wang.

Thanks to my wife Yanfang Ji, and our parents. They give me a happy feeling of family.

In addition, I wish to thank the University of California, Department of Information and Computer Science for providing the UCI Repository of machine learning databases [BM98].

List of Publications

The following is a list of my research papers published in the proceedings of referred international conferences or journals during my PhD study at University of Technology, Sydney.

Referred Journal Papers:

1 Shichao Zhang, Zhenxing Qin, Charles Ling and Shengli Sheng, "Missing is Useful": Missing Values in Cost-sensitive Decision Trees," **IEEE Transactions on Knowledge and Data Engineering**, Vol. 17 No. 12 (2005): 1689-1693.

2 Chengqi Zhang, Zhenxing Qin, Xiaowei Yan, "Association-Based Segmentation for Chinese-Crossed Query Expansion," **IEEE Intelligent Informatics Bulletin** 5(1), 2005: 18-25.

Referred Conference Papers:

3 Zhenxing Qin, Chengqi Zhang and Shichao Zhang, "Missing or absent? A Question in Cost-sensitive Decision Tree," **Proceedings of the Fourth International Conference on Active Media Technology** (AMT006), Jun 2006:

4 Yiming Yang, Qiang Yang, Rong Pan et al. and Zhenxing Qin, "Preprocessing Time Series Data for Classification with Application to CRM," In **Proceedings of the 18th Australian Joint Conference on Artificial Intelligence** (AI 2005), Sydney, Australia, 2005: 133-142.

5 Zhenxing Qin, Chengqi Zhang and Shichao Zhang, "Dynamic Test-sensitive Decision Trees with Multiple Cost Scales," In **Proceedings of International Conference on Fuzzy Systems and Knowledge Discovery** (FSKD-2005), Changsha, China, August 2005: 402-405.

6 Zhenxing Qin, Chengqi Zhang and Shichao Zhang, "Cost-sensitive Decision Trees with Multiple Cost Scales," In **Proceedings of the 17th Australian Joint Conference on Artificial Intelligence** (AI 2004), Cairns, Queensland, Australia, 2004: 380-390.

7 Zhenxing Qin, Li Liu and Shichao Zhang, "Mining Term Association Rules for Heuristic Query Construction," In **Proceedings of 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining**, (PAKDD 2004) Sydney, Australia, May 26-28, 2004: 145-154.

Table of Contents

List of Figures

Chapter 1 Introduction

1.1 Overview of the Thesis

One of the most important abilities of human being is to learn knowledge from previous experience. The interest in computational approaches to learning dates back to the beginnings of artificial intelligence and cognitive science in the mid-1950s. The research field of *machine learning*, which crosses these disciplines, studies the computational processes that underlie learning in both humans and machines since three decades ago. Currently, research on learning and extracting useful information or knowledge from large data stores or sets becomes the mainstream in autonomous learning area, which is usually named as *Data mining* or knowledge-discovery in databases (KDD).

We could formally define the learning as in [Lan96]: *Learning* is the improvement of performance in some environment through the acquisition of knowledge resulting from experience in that environment. An intuitive learning method is *inductive learning*. Inductive learning studies existing examples (called training data) and gives a summary model. Once we meet with similar problems, we could achieve better decisions using the summary model.

This research is about the science of cost-sensitive learning involving multiple cost scales in learning procedure which is a branch of inductive learning. Before turning to the details of my research in cost-sensitive learning, let us set the stage by reviewing the history of the field.

Traditional inductive learning builds classifiers to minimize the expected number of misclassification errors (also known as 0/1 loss). Some inductive learning techniques, such as the decision tree algorithms and naive Bayes, have met with great success in building classification models with the aim to minimize the classification errors [Mit97, Qui93]. Some practice systems have been commonly used in decision-making applications, such as CART and C4.5.

However, there is unbalance on the misclassification errors in some real world application. In other words, sometimes predicting an example of class i into class j might different from predicting an example of class j into class i. Within this scenario, a quantitative measure, noted as *misclassification cost*, is assigned to present the influence of a specific misclassification error.

Definition 1.1 ***Misclassification Cost*** c(i,j) is a value assigned to the misclassification error predicting an example to class i, when it actually belongs to class j.

Suppose there are C classes. In general, we may have a C x C matrix, called ***Cost Matrix***, where the entry e(i, j) is the misclassification cost c(i,j). And the unbalance of misclassification of class i and j is modelled as $c(i, j) \neq c(j, i)$. Assume c(1,2) =1 and c(2,1) = 5, it means misclassifying an example of class 1 to class 2 will cause the same influence as misclassifying five examples of class 2 to class 1. We could find lots of real applications. For example, in medical domain, misclassifying a person with cancer as non-cancer class usually causes more serious problem than the case in reverse.

Much previous inductive learning research has also focused on how to minimize the costs of classification errors. The work is named as 'Cost Sensitive Learning' and categorized as a new branch of inductive learning.

Misclassification error is not the only error in classification problem. In Turney's survey article [Tur00], a whole variety of costs in machine learning are analysed, and the attribute test cost is singled out as one of the least considered. And researchers recently have begun to consider how to involve two or more costs in the same model, such as involving both attribute test costs and misclassification costs [Tur95, GGR02, LYWZ04].

Definition 1.2 ***Test Cost*** of an attribute is the cost incurred for obtaining this attribute's value.

Let us consider the task of diagnosing diabetes [Zub2003]. A doctor may ask the patient a series of questions (such as the health history, family history of medical conditions), perform simple measurements (measuring body mass index, blood pressure) and order lab tests (glucose, insulin). Each measurement or test has an associated cost that is so-called ***test cost*** of this measurement or test. Some test costs are low (i.e., measuring the weight and calculating the body mass index), and some are higher (for example blood test and X-ray test). The diagnostician analyses the results of each selected test and decides whether there is enough information to make a diagnosis or whether more tests are needed. When making a diagnosis, he must take into account the likelihood of each disease and the costs of the misdiagnoses. For example, diagnosing a diabetic patient as healthy can incur costs (such as the cost of aggravating the patient's medical condition); diagnosing a healthy patient as having diabetes can also incur costs (such as the cost of unnecessary treatments).

This dissertation argues for considering both test cost and misclassification cost, or even more costs together. The goal of this dissertation is to study the fundamental properties of attribute test costs and present new algorithmic approaches to combine the two costs together.

Some essential properties of attribute test cost are studied. In our learning problem definition, test cost is combined into misclassification cost by choosing and performing proper tests for a better decision. Why do you choose them and how about the ones that are not chosen? Very often, only part of all attribute values are enough for making a decision and rest attributes are left as "unknown". The values are defined as '*absent values'* or 'absent data' as they are left as unknown purposely for some rational reasons: when the information obtained is considered as enough, or when patients have no money enough to perform further tests, and so on. The "*absent data*" are usually treated as '*missing data*' when left as unknown for unexpected reasons.

Definition 1.3 In a test cost sensitive learning model, very often, some data is left as unknown purposely for some rational reasons, such as enough information obtained, or patients have money enough to perform further tests, etc. This kind of data is noted as ***"absent data".***

Definition 1.3 In a test cost sensitive learning model, some data is left as unknown for unknown for unexpected reasons, such as sensor fault, storage equipment damage, etc. This kind of data is noted as ***"missing data".***

Two novel test cost sensitive models are designed for different real work scenarios. The first model is a *general test cost sensitive learning framework with multiple cost scales*. Previous works assume the test cost and the misclassification cost must be defined on the same cost scale, such as the dollar cost incurred in a medical diagnosis. And they aim to minimize the sum of the misclassification cost and the test cost. However, costs may be measured in very different units and we may meet difficulty with defining the multiple costs on the same cost scale. It is not only a technology issue, but also a social issue. In medical diagnosis, how much money should you assign for a misclassification cost? Sometimes, a misclassification may hurt a patient's life. And from a social point of view, life is invaluable. To tackle this issue, a *target-resource budget learning framework* with multiple costs is proposed. With this framework, we present a test cost sensitive decision tree model with two kinds of cost scales. The task is to minimize one cost scale, called target cost, and keep the other one within specified budgets. To the best of our knowledge, this is the first attempt to study the cost sensitive learning with multiple costs scales.

The second model is based on the assumption that some attributes of an unlabeled example are known before being classified. A test cost sensitive lazy tree model is proposed to utilize the known information and reduce the overall cost. We also modify and apply this model to the batch-test problem: multiple tests are chosen and done in one shot, rather than in a sequential manner in the test-sensitive tree. It is significant in some diagnosis applications that require a decision to be made as soon as possible.

This is the first work to utilize the information hidden in those "absent data" in cost sensitive learning; and the conclusion is very positive, i.e. "absent data" is also useful for decision making. This thesis studies the difference between the ‘absent data’ and ‘missing data’, i.e. ‘absent data’ is trivial and we could just use other data to make a decision; whereas ‘missing data’ may be important for decision making and we should patch up them before performing our learning task. A hybrid lazy decision tree based algorithm is applied to identify the absent data from missing data, and a novel strategy is proposed to help patch the “real” missing values.

Extensive experiments are conducted for evaluating the proposed approaches, and have demonstrated that the work in this dissertation is efficient and useful for many diagnostic tasks involving target cost minimization and resource cost control for obtaining missing information.

1.2 Contributions

The main contributions of this thesis are briefly summarized as follows:

i) The first proposal notices the usage of “absent data” and makes a new conclusion of unknown value problem in cost sensitive learning, i.e. “Absent is useful”. The proposal of evaluating and identifying the missing and absent values with a dynamic tree strategy. And a novel cost sensitive patching model is proposed to patch the “real missing values” [ZQLS05, QZZ06]

ii) The proposal of a new CSL problem when we meet with difficulty defining multiple costs on a certain cost scale. The proposal of “general target – resource budget framework” for this problem by involving two kinds of cost scales into cost sensitive decision tree [QZZ05].

iii) The proposal of a lazy test sensitive decision tree with multiple cost scales for test sensitive decision tree for utilization of known information. And applying the lazy test sensitive decision on the batch testing [FSDK05, QZZ05].

1.3 Organization of the Thesis

The organization of this thesis is as follows. In Chapter 2, some important fundamental concepts, models, approaches, and current research progress related to this research work are briefly overviewed. Chapter 3 addresses the difficulty in defining multiple costs on the same cost scale. A target – resource budget framework is proposed for building a cost-sensitive decision tree involving two kinds of cost scales, that minimizes one kind of cost and control the other in a given specific budget. This chapter also studies the test strategies related to the proposed framework. Chapter 4 studies the lazy test cost sensitive decision tree that aims to utilize known information in examples to be classified. It also studies the batch testing issue and proposes a hybrid lazy tree strategy for batch test selection. Chapter 5 studies the property of performing test to obtain attribute values. This is the first work to propose the usage of absent values and claim usefulness of the "absent" data. Then we study the difference between missing and absent values, and apply the lazy cost sensitive tree to identify the real "missing values". After identifying the absent data, a novel cost sensitive patching model is proposed to patch the "real missing values". Finally, Chapter 6 summarizes the dissertation, provides conclusions derived from the study, and identifies directions for future research in this field.

Chapter 2 Background and Literature Review

This chapter formally introduces the problem of test cost-sensitive classification. Inductive learning techniques have met with great success in building models that assign testing cases to classes and aims to minimize misclassification errors. The CSL is an extension of classic inductive learning, such as supervised learning. We define cost-sensitive learning (CSL) as a procedure of learning diagnostic policies from a set of training examples that minimize a function of multiple costs (such as diagnostic tests and classification errors) and satisfy some constrains on the costs. This formulation is an extension of definition in [Zub2003] that only considers the sum of diagnostic tests and classification errors. Indeed, as in supervised learning, we want to learn a hypothesis predicting the class of new, unseen examples, from a set of labelled training examples. This dissertation focuses on CSL models that are sensitive to both of misclassification cost and test cost. We choose cost sensitive decision tree as the platform for this study. Decision tree is one of the most commonly used techniques in inductive learning. It is with advantage on easy explanation and fast execution, and it is with satisfactory performance on the most of applications. In section 2.1, we briefly introduce the background and basic conceptions of cost sensitive learning and the foundation knowledge of classic decision tree and cost sensitive decision tree. Then we introduce the types of costs in cost sensitive learning. Finally, we discuss the unknown data issue in machine learning and data mining. In Section 2.2, we review relevant literatures from the perspective of costs and their combination. In Section 2.3, we present a paradigm of cost sensitive decision involving both of the misclassification cost and the test cost. Finally we summarize this chapter in last section.

2.1 Background and Definition

2.1.1 Classic Decision Trees

We are confronted with uncertainty every day. To decide how to act, we envision different outcomes of our actions and we plan ahead from each contingency, assessing and weighing the risks and benefits of different courses of action. It is called *sequential decision problem* to make a plan with a set of sequential actions and to choose an action according to the results of previous actions. Decision tree learning abstracts this problem as a learning process from a data table. Each action is expressed as an attribute and performing an action means performing a test on this attribute to obtain its values. The objective is to produce an optimal policy on the order of those attributes.

Examples 2.1: In the domain of credit card application, all applicants could be classified as "high risk" or "low risk". The "high risk" applications would be rejected and "low risk" ones would be approved. The credit card companies, such as Visa card or Master card, may investigate the finance situation of an applicant step by step and finally make a decision. They first check the applicant's annual income, if his annual income is less then 30K, he will be rejected directly because his income doesn't reach the requirement. If his annual income is over 30K, the company will check his criminal record for security reason. The application will be approved if he has no criminal record, otherwise it will be rejected.

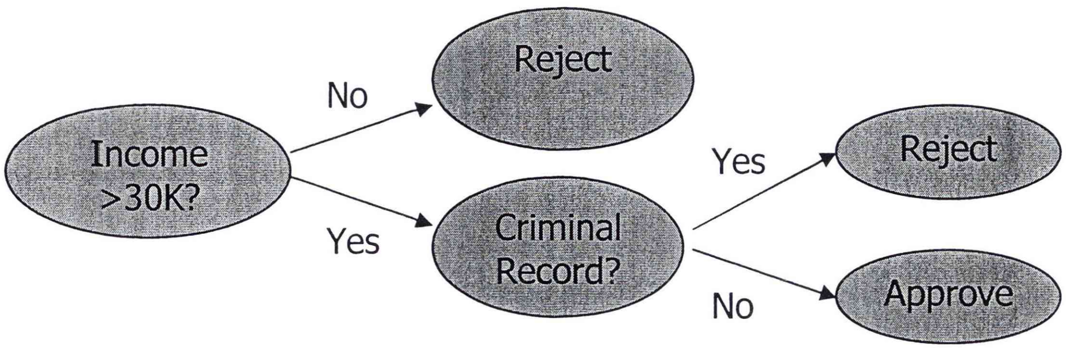


Figure 2.1 An example of Decision Tree on Credit Card Application.

The core issue of decision tree learning is to select proper test at each node, such as in example 2.1, we decide to check the applicant's annual income before checking his criminal record. Many criteria are designed for the selection. Entropy-based selection measure is one of the most commonly used criteria.

To describe the entropy-based selection measure, we follow the notation in [CT991, FYK96]. Let *Y* be a discrete random variable with range y; the entropy of *Y*, sometimes called the information of *Y*, is defined as

$$H(Y) = -\sum_{y \in Y} p(y) * \log(p(y)) \qquad (2.1)$$

where 0 log 0 = 0 and the base of the log function is usually two so that entropy is expressed in bits. The entropy is always non-negative and measures the amount of uncertainty of the random variable *Y*. It is bounded by log |y| with equality only if *Y* is uniformly distributed over y.

The conditional entropy of a variable *Y* given another variable *X* is the expected value of the entropies of the conditional distributions averaged over the conditioning random variable:

$$H(Y;X) = -\sum_{x \in X} p(x) H(Y \mid X = x) \qquad (2.2)$$

$$= -\sum_{x \in X} p(x) \sum_{y \in Y} p(y \mid x) \log p(y \mid x) \qquad (2.3)$$

$$= -\sum_{y \in Y} \sum_{x \in X} p(x, y) \log p(y \mid x) \qquad (2.4)$$

Note that H (Y|X) $\neq$ H(X|Y).

The mutual information of two random variables *Y* and *X*, sometimes called the information gain of *Y* given *X*, measures the relative entropy between the joint distribution and the product distribution:

$$I(Y; X) = \sum_{y \in Y} \sum_{x \in X} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} \qquad (2.5)$$

$$= \quad \mathrm{H(Y) - H(Y \mid X)} \qquad (2.6)$$

The mutual information is symmetric, i.e., I*(Y;X)* = *I(X;Y)*, and non-negative [CT91]. As can be seen from Equation 2.6, the mutual information measures the reduction in uncertainty in *Y* after observing *X*. Given a set of instances, the above quantities can be computed by using the empirical probabilities, with the variable Y representing the class labels and *X* a given feature variable.

The test selection step of common decision tree algorithms is implemented by testing the mutual information (or a similar measure) for each feature *X* with the class label Y and picking the one with the highest value (highest information gain).

Many decision tree algorithms, such as C4.5 and CART, have a post-processing step that prunes the tree to avoid over-fitting. Readers are referred to [Qui93, BFOS84] for the two most common pruning mechanisms.

Formally, given a sample of labeled examples (x; y) drawn from a distribution D(x; y), where x is a vector of attributes and y is its label, the task is to learn a *hypothesis* h that labels x with the most likely class. The predicted class h(x) is denoted by y'. The distribution D(x; y) from which the labeled examples are drawn can be factored into two probability distributions, a class probability P(y) and a conditional probability P(x|y).

The attributes can be symbolic or numeric (discrete or continuous). The labels can be discrete (in which case the task is called classification, and the labels are called classes) or continuous (in which case the task is called regression, for example, predicting the temperature in a furnace).

Our CSL framework assumes the attributes are numeric and the labels are discrete, so it focuses on classification tasks. The labels are called the observed classes.

In traditional classification tasks, the goal is to find a hypothesis *h* that minimizes the expected number of *misclassification* errors, that is, the expected number of examples incorrectly classified:

$$Min_{\,E(x;\,y)\sim D}\,[L(h(x);y)] = Min\,\Sigma_{(x,\,y)}\,D(x;\,y)L(h(x);y); \qquad (2.7)$$

where the loss function L(h(x);y) is 0 when h(x) = y and =1 otherwise; These classifiers are also known as minimizing the expected 0/1 loss, and their underlying assumption is that misclassification errors have the same cost, and, in addition, no attention is paid to attribute costs.

2.1.2 Classic Cost Sensitive Decision Tree

The Cost Sensitive Learning is an extension of classic inductive learning for the unbalance in misclassification errors, i.e. misjudging a case with class i as class j is different from misjudging a case with class j as class i.

Example 2.2: Considering the credit card application problem in example 2.1, we also need to classify the customers as “high risk” or “low risk”. But different from traditional classification here, misclassifying a “high risk” application as “low risk” might lead to a fraud easily and cause average $4000 loss; contrarily misclassifying a “low risk” application as “high risk” means we lose a good customer and will lose average $1000 benefit from the customer.

From the example above, the two kinds of misclassification errors would incurs different cost and we should pay more attention on the high one. Traditional inductive learning treats the two misclassification errors as the same, so it is no longer proper for this problem. It is motivated to extend the traditional inductive learning to cost sensitive learning.

Indeed, as in supervised learning, we want to learn a hypothesis predicting the class of new, unseen examples, from a set of labelled training examples. But our objective function is cost-sensitive, and subject to its minimization, we want to learn in which order to perform the diagnostic tests followed by classification actions. Our CSL framework assumes the attributes are numeric and the labels are discrete, so it focuses on classification tasks. The labels are called the observed classes. Here we briefly introduce the basic conceptions of cost sensitive learning.

2.1.3.1 Properties of cost matrix

Cost matrix is a matrix used to describe the unbalance of misclassification errors, where an element a(i ,j) is the cost of misjudging a case as class i but its true class is j. A cost matrix C always has the structure as in Table 2.1 when there are only two classes: In Table 2.2, there are only two classes (0 and 1), we call a positive example one whose class is y = 1 and a negative example one whose class is y = 0. It helps to think of this in terms of medical diagnosis. A patient is “diabetes positive" if he has the disease (y = 1) and “diabetes negative" if he does not (y = 0). The classes assigned by the hypothesis can be correct or not, so we talk about ***“true positives***" (noted as *TP*) and ***“true negatives***" (noted as *TN*), when the predictions match the observed classes, and ***“false positives***” (noted as *FP*) and “false negatives" (noted as *FN*) when they do not. Let y^ = h(x) be the class predicted by hypothesis h. Then a false positive is an example (x; y) where y^ = 1 and y = 0 (a healthy patient was diagnosed with diabetes). Similarly, a false negative example was assigned class y^ = 0 when in fact its observed class is y = 1 (a sick patient was diagnosed to be healthy).

	Actual y=0	Actual y=1
Predict y=0	true negatives	false negatives
Predict y=1	false positives	true positives

Table2. 1 Notations for the examples' predictions versus their observed classes. The class $y = 1$ is interpreted as having the disease, and $y = 0$ as not having it.

After a hypothesis is learned, we want to see how good it is at predicting the classification of new, unseen examples. The strategy is to divide the data into two sets, a training set and a test set. Learning is done on the training set; then the hypothesis is evaluated on the test set. Because we know the observed labels of the test examples, we can compare them to the predicted labels of the hypothesis and compute the number of errors (false positives and false negatives).

	Actual Bad	Actual Good
Predict Bad	0	1
Predict Good	5	0

Table2. 2 Example of cost matrix of German credit dataset

Cost-sensitive learning is an extension of the classification task of supervised learning, and motivated from the unbalance in misclassification errors that are usually measured with a cost matrix. For example, a cost matrix of the so-called German Credit dataset given by [BFOS84] is shown in table 2.2. The cost matrix is used to evaluate the credit of people who apply for a loan from a bank. "Actual good" means that a customer would repay a loan while "Actual bad" means that the customer would default. Then a classifier is built to predict if an applicant is worth to be given a loan where "predict good" is to approve the loan, otherwise to deny the loan.

2.1.3.2 Making decisions based on a cost matrix

Given a specification of costs matrix C, an example should be predicted to have the class that leads to the lowest expected cost, where the expectation is computed using the conditional probability of each class given the example. Mathematically, let the (i,j) entry in a cost matrix be the cost of predicting class i when the true class is j. If i = j then the prediction is correct, while if i ≠ j the prediction is incorrect. Assume there are n class labels, then optimal prediction for an example x is the class i that minimizes

$$L(x,i) = \sum_{j=1}^{n} p(j \mid y) C(i,j) \qquad (2.8)$$

For each i, L(x, i) is a sum over the alternative possibilities for the true class of i. In this framework, the role of a learning algorithm is to produce a classifier that for any example can estimate the probability P(j | x) of each class j being the true class of x.

Let's see an example in [Elk01], in the two-class case, the optimal prediction is class 1 if and only if the expected cost of this prediction is less than or equal to the expected cost of predicting class 0, i.e. if and only if

$$P(j=0\mid x)C_{10}+P(j=0\mid x)C_{11}\le P(j=0\mid x)C_{11}+P(j=0\mid x)C_{10} \tag{2.9}$$

which is equivalent to

$$(1-p)C_{10}+pC_{11}\le(1-p)C_{11}+pC_{10} \tag{2.10}$$

if we note p = P(j = 1 |x) If this inequality is in fact an equality, then predicting either class is optimal.

The threshold for making optimal decisions is p* such that

$$(1-p^*)C_{10}+p^*C_{11}=(1-p^*)C_{11}+p^*C_{10}$$

Rearranging the equation for p*leads to the solution

$$p^*=\frac{C_{10}-C_{00}}{C_{10}-C_{00}+C_{01}-C_{11}}$$

According to [Elk01], the optimal prediction is class 1 if and only if p ≥ p*. This formula for p* shows that any 2x2 cost matrix has essentially only one degree of freedom from a decision-making perspective, although it has two degrees of freedom from a matrix perspective. The cause of the apparent contradiction is that the optimal decision-making policy is a nonlinear function of the cost matrix.

2.1.3 Types of Cost in Cost Sensitive Learning

Misclassification error is not the only error in classification problem. In Turney's survey article [Tur00], a whole variety of costs in machine learning are analyzed. The costs of different types of errors are often very different. We will give some details of the various costs. The first two types of costs are the misclassification costs that are the costs incurred by misclassification errors and the test costs that are the costs incurred for obtaining attribute values. According to this work, "cost" should be interpreted in its most abstract sense. Cost may be measured in many different units, such as monetary units (dollars), temporal units (seconds), or abstract units of utility (utils). In medical diagnosis, cost may include such things as the quality of life of the patient, in so far as such things can be (approximately) measured. In image recognition, cost might be measured in terms of the CPU time required for certain computations. We briefly list the types of cost as follows:

2.1.3.1 Costs of Test

Using Turney's terminology for conditional test costs [Tur00], the measurement cost of attribute x n may depend on

- prior test selection (e.g., blood tests can share a common cost of collecting the blood).
- the results of prior tests (e.g., drawing blood from a newborn is more costly than from an adult; in this case, the result of a previous test "observe patient age" influences the cost of the next test "draw blood").

- the outcome of x_n (e.g., in computer network diagnosis [LNHFH03], doing a "ping" to measure the round-trip-time to a host is very fast if the host is reachable but waits 20 seconds for a timeout if the host is down or not reachable).
- the class (e.g, tests can become more expensive for patients in critical medical condition).

In general, measurement costs can depend on the action performed x_n, the current state s (history of prior measurements and their values), the resulting states s' (since s' = s U {x_n = v}, this is a dependency on the outcomes of x_n), and the class of the example y, so the most general form of the cost function is C(s, x_n, s', y). In order to reason with complex test costs, we would first need to acquire them, either from training data or from being told. Note that we cannot learn cost dependencies from our existing, order-free, training data. Once the cost model is known, we can easily incorporate more complex test costs in our framework than the current test costs C(s; x_n).

2.1.3.2 Cost of Misclassification Errors

Suppose there are C classes. In general, we may have a C x C cost matrix MC (i, j), where the element in row i and column j specifies the cost of assigning a case to class i, when it actually belongs to class j. Typically (but not necessarily) the cost is zero when i equals j. In a minor variation on this approach, we may have a rectangular matrix, where there is an extra row for the cost of assigning a case to the unknown (or "too-difficult-for-this-learner") class.

The cost of misclassification could be constant or conditional. The constant cost means using a constant cost matrix (the values of a cell in the cost matrix are constant) for all cases. This is the most commonly investigated type of cost; for example, see

[BFOS84, HHB74]. The conditional cost means the costs may be conditional on the circumstances. The cost of a misclassification error may depend on the nature of the particular case, time of classification, or other cases. For example, in detection of fraud, the cost of missing a particular case of fraud will depend on the amount of money involved in that particular case [FP96, FP96].

In our experimental studies, we considered the simple case where test costs are constant (they depend only on the attribute x_n, $C(x_n)$), and misclassification costs are constant as well (they do not depend on examples, just on the predicted and observed class). But we can incorporate both attribute and misclassification costs of these more complex forms in our CSL framework.

2.1.3.3 Cost of Teacher

Suppose we have a practically unlimited supply of unclassified examples (i.e., cases, feature vectors), but it is expensive to determine the correct class of an example. For example, every human is a potential case for medical diagnosis, but we require a physician to determine the correct diagnosis for each person. A learning algorithm could seek to reduce the cost of teaching by actively selecting cases for the teacher. A wise learner would classify the easy cases him/herself and reserve the difficult cases for his/her teacher.

If a learner has no choice in the cases that he/she must classify, then it can only rationally determine whether he/she should pay the cost of a teacher when he/she knows the cost of misclassification errors. A rational learner would, for each new case, calculate the expected cost of classifying the case by him/herself versus the cost of asking a teacher to classify the case. This scenario can be handled by using a rectangular cost matrix, as we discussed in Section 1.2.

In a more interesting scenario, the learner can explore a (possibly infinite) set of unclassified (unlabelled) examples and select examples to ask the teacher to classify. This kind of learning problem is known as active learning. In this scenario, we can rationally seek to minimize the cost of the teacher even when we do not know the cost of misclassification errors, if we assume that asking the teacher costs more than a correct classification (otherwise you would always ask the teacher) but less than an incorrect classification (otherwise you would never ask the teacher). However, we may be able to make better decisions if we have more information about the cost of misclassification errors.

2.1.3.4 Other Costs

This dissertation mainly focuses on the test cost and misclassification cost. Cost of Teacher is going to be considered in future work. There are other four kinds of costs. Their names are simply listed as follows. Readers are referred to [Tur00] for more details.

- **Cost of Intervention**
- **Cost of Unwanted Achievements**
- **Cost of Computation**
- **Cost of Cases**
- **Human-Computer Interaction Cost**

2.1.4 Unknown Data in Machine Learning and Data Mining

With the advancement in computer and information technologies, the ever-growing data sets stored in large amount of databases and data warehouses. There is abundant and precious information (knowledge) hidden in the huge amount of data. However, in most databases and data warehouses, raw data are not ready to be processed directly by data mining tools because they may contain a great many irrelevant, inconsistent, or missing data items. To be useful to learning or mining, the databases need to undergo pre-processing, in the form of data cleaning and data transformation [Lar05, ZZW04, HK00]. Unknown data is the most common issue that some fields are not recorded for some reasons. From our point of view, unknown data could be classified into two categories: *absent* and *missing*. If the data is left unknown purposely we call it absent data, otherwise we call it missing data. In traditional machine learning and data mining, unknown values are all treated as missing values, i.e. the data is unknown for some unexpected reasons. In this research, we argue the need for learning from data with absent values and we discuss more details in chapter 6 and 7. In this section, we only discuss the essential properties of missing data.

Missing data handling is a main task in the data preparation phase. In most cases, missing data should be pre-processed (recovered) so as to allow the whole data set to be processed by a data-mining tool or a learning algorithm. Now we discuss the properties and categories of "missing" itself from the perspective of statistics.

There are several reasons why the data may be missing. They may be missing because equipment malfunctioned, the weather was terrible, or people got sick, or the data were not entered correctly. While attributes in most data sets can be distinguished in categories of randomly distributed or non-randomly distributed, the missing data can also be distinguished in these two categories: (1) non-randomly distributed, and (2) randomly distributed [HZ02]. That is, the mechanisms underlying the situations of certain data being missing can be characterized as either random or non-random. But this randomness is by no means related to the randomness of the attribute in the original data set, or at least we do not assume that in this study.

The issue of missing values (or missing data) has been studied extensively in the statistical and machine learning literature. According to the missing data mechanisms, statisticians have identified three classes of missing data [LR87]: *missing completely at random (MCAR)*, *missing at random (MAR)*, and *not missing at random (NMAR)*. MCAR is when the probability of missing a value is the same for all variables; MAR is when the probability of missing a value is only dependent on other variables; and NMAR is when the probability of missing a value is also dependent on the value of the missing variable. MAR has received most attention, for which various "imputation" methods have been designed to predict the missing values before building models.

Missing data are a part of almost all researches, and we all have to decide how to deal with it from time to time. There are a number of alternative ways of dealing with missing data, and this section is an attempt to outline those approaches.

Currently, there are three approaches to deal with missing fields: mark, filtering and imputation. The mark method is to mark all the unknown values by a special symbol, usually called null value which means the values is existent but not recorded. It exactly does nothing about the missing fields and leaves the data imperfections to data mining algorithms. Many algorithms in the machine learning are robust enough to handle the special values, such as C4.5.

Filtering simply discards those data instances with missing fields and only uses the rest data for data mining. This method often results in a substantial decrease in the sample size available for the analysis. This is certainly not satisfactory as it may result in wasting of data. In particular, it always assumes that data are "missing at random", otherwise it may leads to biased estimates on true patterns in data.

Imputation method is currently commonly used, which assigns values to these missing fields based on some criteria. Many criteria are proposed for variant domains, such as statistics estimation, correlation analysis, and association among attributes, etc. In machine learning, the missing value issue has been dealt with mostly in decision tree learning and rule learning. Various imputation methods have also been tried, such as imputation by the most common value [CN89], clustering [CS95], and other learning models [BFOS84]. In C4.5 [Qui89, Qui93] a different approach is used in which a test example with missing values is distributed into branches probabilistically (see Section 3.4). Comparison of various imputation methods has also been published [LHS99]. The approaches we discuss in this thesis do not impute any missing values, as it is regarded as unnecessary for cost-sensitive learning that also considers the test costs.

We only gave an outline of dealing with missing data in this section. You can find a very thorough book-length treatment of the issue of missing data in [LR87] .A shorter treatment can be found in [All02].

2.2 Literature Review for Cost-sensitive Learning and Cost Combination

In this section, we review the literatures of cost and cost combination in cost sensitive learning. Machine learning has tackled several different settings for the classification problem. Here we categorize the existing classification from the perspective of cost type and their combination. There are two overall categories: "classifiers for single cost" and "classifiers for multiple costs" according to the number of cost involved. The first category includes classic classifiers for misclassification errors, attribute cost and misclassification cost. Those classifiers are advanced and well developed in last three decades. Classifiers in the second category try to involve multiple costs into a classifier. Most of the classifiers involving multiple costs are proposed in the last 5 years and they represent a latest research direction in this area.

This dissertation will mainly focus on classifiers in second category, especially the classifiers sensitive to both attribute costs and misclassification costs.

2.2.1 Classifiers for Single Cost

1. **Classifiers minimizing 0/1 loss**. This has been the main focus of machine learning, from which we mention only CART [BFOS84] and C4.5 [Qui93]. These are standard top-down decision tree algorithms. C4.5 introduced the information gain as a heuristic for choosing which attribute to measure in each node. CART uses the GINI criterion.

Weiss et al. [WGT90] proposed an algorithm for learning decision rules of a fixed length for classifications in a medical application; there are no costs (the goal is to maximize prediction accuracy). Their paper also defines the commonly used medical terms of sensitivity and specificity of tests from a machine learning perspective.

2. **Classifiers sensitive only to attribute costs:** Norton [Nor89], Nunez [Nun91] and Tan [TS90, Tan93]. The splitting criterion of these decision trees combines information gain and attribute costs. These policies are learned from data, and their

objective is to maximize accuracy (equivalently, to minimize the expected number of misclassification errors) and to minimize expected costs of attributes.

A related problem is the *test sequencing problem* [PA90], from electronic systems testing. Pattipati and Alexandridis pointed out that "the test sequencing problem belongs to the general case of binary identification problems that arise in botanical and zoological field work, plant pathology, medical diagnosis, computerized banking and pattern recognition." The objective of the test sequencing problem is to unambiguously (deterministically) identify the system state (either one of the faulty states, or the fault-free state) by performing tests with minimum expected total cost. The assumptions are that only one of the system states occurs (or equivalently, the faults are mutually exclusive), the probability distribution over the system states is given and so is the binary diagnostic matrix (which tells if a test detects a fault or not). The test sequencing problem is a simplified version of the cost-sensitive classification problem, because faults are identifiable with probability 1.0, and therefore there are no misclassification costs.

3. **Classifiers sensitive only to misclassification costs:** Breiman and al. [BFOS84], Hermans et al. [HHB74], Gordon and Perlis [GP89], Pazzani et al. [PMMA94], Knoll et al. [KNT94], Fawcett and Provost [FP97], Gama [Gam00], Margineantu [Mar01], Zadrozny and Elkan [ZE01]. This problem setting assumes that all data is provided at once, therefore there are no costs for measuring attributes and only misclassification costs matter; this is not a sequential decision making problem. The task is to minimize the expected misclassification costs.

This work can be further divided depending on at which point in the learning process the knowledge about misclassification costs becomes available:

(a) **Misclassification costs known during the learning of classifiers:** CART [BFOS84], MetaCost [Dom99], post-pruning of decision trees using misclassification costs (Bradford et al. [BKKB98], Kukar and Kononenko [KK98], Webb [Web96]).

(b) **Misclassification costs not known until execution time.** Two strategies are employed. The first one learns cost-insensitive classifiers with improved conditional class probabilities *P(y\x),* then classifies each test example *x* into the class with the minimum expected cost:

$$\hat{y}_{opt}(x) = \arg\min_{y^\wedge} \sum_{y} p(y \mid x) L(y^\wedge, y)$$

This approach includes logistic regression, Friedman and Stuetzle's projection pursuit regression [FS81], Naive Bayes, Domingos and Provost's B-PETs [DP00], Margineantu and Dietterich's B-LOTs [MD02].

The second strategy learns a range of operating points on an ROC curve. When costs become known at execution time, an operating point is chosen (Provost and Fawcett's ROC convex hull [PF97, PF01]).

2.2.2 Classifiers for Multiple Costs

4. Classifiers sensitive to both attribute costs and misclassification costs.

Currently, researchers have begun to consider both test and misclassification costs [Tur95, GGR02, LYWZ04]. **The task is to minimize** the expected total cost of tests and misclassifications.

Peter Turney [Tur95] developed a learning system, called ICET, a cost-sensitive algorithm that employs genetic search to tune parameters used to construct decision trees. Each decision tree is built using Nunez' criterion (described in Section 3.2), which selects attributes greedily, based on their information gain and costs. Turney's method adjusts the test costs to change the behavior of Nunez' heuristic so that it builds different trees. These trees are evaluated on an internal holdout data set using the real test costs and misclassification costs. After several trials, the best set of test costs found by the genetic search is used by the Nunez' heuristic to build the final decision tree on the entire training data set. Because Turney simply modifies the attribute selection in C4.5 to add attribute costs when implementing the Nunez' criterion, his algorithm can deal with continuous attributes and with missing attribute values. [Tur95] is also a seminal work laying the foundations of cost-sensitive learning with both attribute costs and misclassification costs. Turney compares his algorithm with C4.5 and with algorithms sensitive only to attribute costs (Norton, Nunez and Tan). He does not compare ICET with algorithms sensitive to misclassification costs only, because in his experiments he used simple misclassification cost matrices (equal costs on diagonal, equal costs off diagonal) which make algorithms sensitive only to misclassification costs equivalent to minimizing 0/1 loss. ICET outperformed the simpler greedy algorithms on several medical domains from the UCI repository.

In [ZD02], the cost-sensitive learning problem is cast as a Markov Decision Process (MDP), and an optimal solution is given as a search in a state space for optimal policies. For a given new case, depending on the values obtained so far, the optimal policy can suggest a best action to perform in order to both minimize the misclassification and the test costs. While related to other work, their research adopts an optimal search strategy, which may incur very high computational cost to conduct the search.

Similar in the interest in constructing an optimal learner, Greiner, Grove and Roth [GGR02] studied the theoretical aspects of active learning with test costs using a PAC learning framework. It is a theoretical work on a dynamic programming algorithm (value iteration) searching for best diagnostic policies measuring at most a constant number of attributes. Their theoretical bound is not applicable in practice, because it requires a specified amount of training data in order to obtain close-to-optimal policies.

Ling, Yang, Wang and Zhang [LYWZ04] proposed a new method for building and testing decision trees involving misclassification cost and test cost. The task is to minimize the expected total cost of tests and misclassifications. It assumes a static cost structure where the cost is not a function of time or cases. It also assumes the test cost and the misclassification cost have been defined on the same cost scale, such as the dollar cost incurred in a medical diagnosis. In the later part of this section, we will provide some details of this work because most of our work is based on this work.

Following the work in [LYWZ04], further research has been done by us and our collaborators. Qin and Zhang [QZZ04] proposed a general framework for involving multiple costs in different cost scales. The task is to minimize one cost scale and control other cost scales in specified budgets. Chai and Ling [CDYL04] proposed a test cost sensitive naïve Bayesian network. C. Ling, Q.Yang have done much work in test strategies in test cost sensitive learning [SLY05, SLNZ06, SL06, LSY06], and they aim to seek the best test attribute set for decision making. Zhang, Qin, Ling [ZQLS05] consider the cost sensitive learning in data with missing value and conclude that some data are left as unknown in domain of test cost sensitive learning and could be useful for decision. And Qin and Zhang [QZZ06] formally note those data as "absent data" and propose a new algorithm to distinguish them from real missing data.

5. Classifiers sensitive to misclassification costs and other costs.

A. Arnt and S. Zilberstein [AZ05] try to build a model to manage the tradeoff between decision making time and accuracy. And as we know, C. Ling has submitted a paper to IEEE TKDE for combining teaching cost and misclassification cost. In their paper, each training example is with a specified cost, and learning model need to "buy" a proper number of training examples for decision making. The task is to minimize the overall cost of teaching and misclassification cost. It is still a challenge to combining multiple costs in cost sensitive.

2.3 A Cost Sensitive Decision Tree Involving both of Test and Misclassification Cost

Ling, Yang, Wang and Zhang proposed a new method for building and testing decision trees that minimizes the sum of the misclassification cost and the test cost [LYWZ04]. It assumes a static cost structure where the cost is not a function of time or cases. It also assumes the test cost and the misclassification cost have been defined on the same cost scale, such as the dollar cost incurred in a medical diagnosis. We will simply introduce the tree building based on single cost scale as the following:

To build a decision tree we need two main operations starting from the root node:

#1. *Evaluate unused attributes for current node according to specific splitting criteria; if no one satisfies the splitting criteria, label the node as a leaf according to a specific leaf marking criteria, otherwise perform operation #2;*

#2. *Select the best attribute to split the node, and recursively perform operation #1 and #2 on all of this node's children nodes*

From the two operations above, we need to define two criteria: one for leaf marking and the other for evaluating candidate splitting attributes.

1. Criteria for labeling a leaf

To minimize the total target cost, at each leaf, the algorithm labels the leaf as either positive or negative (in a binary decision case) as following criteria: Assume FN is the cost of *false negative* and FP is the cost of *false positive; i*n each candidate node, there is a set of P positive and N negative examples respectively to be further classified by possibly building a sub-tree. And if there is no sub-tree is built, we will label the node as positive if $P \times FN > N \times FP$, otherwise it would be labeled as positive. In other words, a class is chosen for the leaf if it is with minimal total misclassification cost Min $\{N \times FP, N \times FP\}$.

For easier discussion in this section, assuming the $P \times FN > N \times FP$, then $T = \text{Min}\{N \times FP, N \times FP\} = N \times FP$

2. Criteria for evaluating candidate splitting attributes

Now let's define the criteria for evaluating candidate splitting attributes. Suppose that an attribute A with a test cost C1 is considered for a potential splitting attribute. Assume that A has two values, and there are P1 and N1 positive and negative examples with the first value, P2 and N2 positive and negative examples with the second value, and P0 and N0 positive and negative examples with A's value unknown. Then the total test cost here would be

$$(P1+N1+P2+N2) \times C1$$

(i.e., cases with unknown attribute values do not incur test costs). Assume that the first branch will be labeled as positive (as $P1 \times FN1 > N1 \times FP1$), and the second branch will be labeled as negative, then the total misclassification cost of the two branches would be

$$N1\times FP1+P2\times FN1$$

As we have discussed earlier in this section, examples with the unknown value of *A* stay with the attribute *A*, and we have assumed that the original set of examples is labeled as positive. Thus, the misclassification cost of the unknowns is N0×FP. The total cost of choosing A as a splitting attribute would be:

$$T_A = (P1+N1+P2+N2)\times C1 + N1\times FP1 + P2\times FN1 + N0\times FP1$$

If $T_A < T$, where $T = N\times FP1$, then splitting on A would reduce the total cost of the original set, and we will choose such an attribute with the minimal total cost as a splitting attribute. We will then apply this process recursively on examples falling into branches of this attribute. If $T_A \geq T$ for all remaining attributes, then no further sub-tree will be built, and the set would become a leaf, with a positive label.

A1	**A2**	**A3**	**A4**	**A5**	**A6**	**FP/FN**
50	50	50	50	50	20	800/600

Table2. 3 Test and misclassification costs set for Ecoli dataset.

We only simply show you a concrete sample cost list and corresponding tree here in Table 2.3 and Figure 2.1 [LYWZ04]. The ratio in the parentheses of each node means the positive example number VS negative example number, such (230:102) in root node.

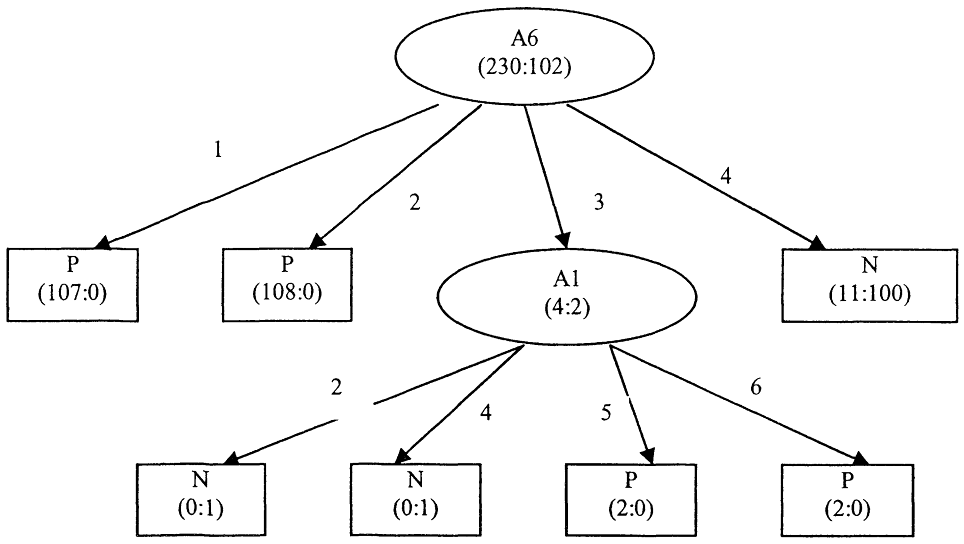


Figure 2.2 A decision tree built from the Ecoli dataset (costs are set as in Table 2.3).

After the tree in Figure 2.2, we use it to predict the class of examples in test set. According to authors' assumption, test examples may contain partial known information before our prediction. A test example is shown in Table 2.4, where the true values are in parenthesis and could be obtained by performing the tests.

A1	A2	A3	A4	A5	A6	Class
? (6)	2	? (1)	2	2	? (4)	P

Table2. 4 An example testing case with several known values.

From the example above, we can see that attribute A2, A4, A5 are known before prediction. However, the tree in Figure 2.2 is built based on the assumption that their values are unknown before test. And in order to predict the class such kind of testing examples with the minimal total cost, four testing strategies were studied. We will briefly introduce them as follows, noted as M1 to M4. When meeting with an unknown value in test example:

The strategy M1, called Optimal Sequential Test (OST), performs extra tests on the unknown values. It uses the tree built with the minimal cost to decide what tests must be performed in sequence.

The strategy M2 stops at the node if its value is unknown, and uses the ratio of positive and negative examples in that (internal) node to predict the testing example (recall that these ratios are calculated based on training cases which also have unknown values at this node).

The third strategy M3 uses the C4.5's strategy in dealing with missing values by choosing a value according to the probabilities of the attribute's all values. Instead of stopping at the node whose attributes value is unknown in the testing case, this strategy will "split" the testing case into fractions according to the training examples, and go down all branches simultaneously.

The fourth and final strategy M4 ignores the attributes with unknown values and uses rest attributes to build a new tree for the test sample.

2.4 Summary

This chapter has formalized the cost-sensitive learning problem by using terminologies from supervised learning. We borrowed the format of the data (sets of labeled examples) and the task of predicting the class from supervised learning. We identified restrictions of our framework and discussed related work.

The following chapters delve into the detailed issues of test sensitive learning algorithms. We firstly give a formal description for test sensitive learning in [LYWZ04] that defines two costs into same cost scale. Based on the description of the test sensitive learning with single cost scale, we then propose a general framework for cost sensitive decision tree with multiple cost scales and discuss its essential properties. Existing cost sensitive learning framework is only a special case of our framework.

Chapter 3 Test Cost Sensitive Decision Trees with Multiple Cost Scales

Recently, researchers have begun to consider both test and misclassification costs in cost sensitive learning. Previous works assume the test cost and the misclassification cost must be defined on the same cost scale. The task is to minimize the expected total cost of tests and misclassifications. However, sometimes we may meet difficulty in defining the multiple costs on the same cost scale. In this chapter, we address the problem by building a cost-sensitive decision tree by involving two kinds of cost scales, which minimizes the one kind of cost and control the other in a given specific budget. The major part of this chapter is published in AI'04 [QZZ04]. The rest of the chapter is organized as follows: In Section 3.1, we first discuss the motivation of this work. In Section 3.2, we simply introduce the tree-building algorithm based on single cost scales. In Section 3.3, we proposed a general framework for involving two cost scales and then discuss the new issues and properties as involving resource control on decision tree. In Section 3.4, we proposed two tree building strategies with multiple cost scales. In Section 3.5, we consider several testing strategies and analyze their relative merits. Finally, we present our experimental results in Section 3.6 and conclude the work with a discussion of future work in Section 3.7.

3.1 Motivation

Misclassification error is not the only error in classification problem. Numbers of different types of misclassification errors are listed in [Tur00]. Currently, much work is reported in researches considering two more costs together in cost sensitive learning, especially both of test and misclassification costs [D95, GGR02]. The task is to minimize the expected total cost of tests and misclassifications. Ling, Yang, Wang and Zhang proposed a new method for building and testing decision trees that minimizes the sum of the misclassification cost and the test cost [LYWZ04]. It assumes a static cost structure where the cost is not a function of time or cases. It also assumes the test cost and the misclassification cost have been defined on the same cost scale, such as the dollar cost incurred in a medical diagnosis.

However, in practical applications, Cost may be measured in very different units. Sometimes we may meet difficulty in defining the multiple costs on the same cost scale. It is not only a technology issue, but also a social issue. For example, in medical diagnosis domain, a misdiagnosis cost is defined as percentage (misclassification rate), the (medical) test is defined as monetary unit (test fee - dollars). We convert the two scales into same scale if we want to use existing learning models that could only handle single cost scale. However, how much money should you assign for a misclassification cost? Sometimes, a misclassification may hurt a patient's life. And from the social point of view, life is invaluable. So we need to involve both of the two cost scales in the same learning model.

On the other hand, assume the two scales above, percentage and monetary unit, are included in a learning model. The best case is to minimize both of them, i.e. getting best diagnosis precision and spending least money. However, it is not realistic to expect to minimize both of them in real world applications. In this case, a trade-off between two costs is necessary. For each individual, he/she may pay more attention to a specific cost scale. A millionaire prefers a minimal diagnosis mistake (it means minimal misclassification cost), and he would like to pay much more money for more detailed tests. But someone else can accept a tolerant misclassification cost with controlling the diagnosis fee in a specific budget (such as the insurance cover limit).

In the next section, a new general framework involving multiple cost scales is proposed to tackle the problems above, which minimizes the one kind of cost while controlling the other in a given specific budget.

3.2 A General Framework for Learning with Multiple Cost Scales

In this section, we give a formal definition for a general framework for learning with multiple cost scales. It is extended from current test cost sensitive learning framework, just like the later is extended from classic cost sensitive learning framework. Here, we briefly describe formal definitions of three frameworks and show their correlation as well.

3.2.1 Classic cost sensitive learning framework

Let's recall the goal of classic cost sensitive learning for misclassification cost in section 2.1.2. Assume there are n class labels in a learning task. An $n \times n$ cost matrix is given. The optimal prediction for an example x is the class i that minimizes

$$L(x,i) = \sum_{j=1}^{n} p(j \mid y) C(i,j) \qquad (3.1)$$

Where *C(i ,j)* is the misclassification cost given by entry *(i,j)* of cost matrix; *P(j | x)* is the probability of each class j being the true class of x. For each *i*, *L(x, i)* is a sum over the alternative possibilities for the true class of i. In this framework, the role of a learning algorithm is to produce a classifier that for any example can estimate the probability *P(j | x)*.

3.2.2 Test cost sensitive learning framework with single cost scale

Current test cost sensitive learning framework combines both of the misclassification cost and test cost together. It aims to minimize the sum of the two kinds of costs. It assumes the test cost and the misclassification cost have been defined on the same cost scale, such as the dollar cost incurred in a medical diagnosis.

When we combine the test cost and misclassification cost into the classification model above, we need to add extra test costs in the formula 3.1, i.e. total cost of all tested attributes for making a decision. Assume there are *m* attributes for test and each attribute *k* is with a test cost t_k, the cost set is noted as T= {t_1, t_2, ... t_m}. Following the formula 3.1, the optimal prediction for an example x in test cost sensitive learning is class *i* that minimizes

$$L'(x,i) = \sum_{j=1}^{n} p(j \mid y) C(i,j) + \sum_{k=1}^{m} p(k) * t_k \qquad (3.2)$$

Where t_k is the test cost of attribute *k*; *p(k)* is the probability of performing a test for the value attribute *k* while making the decision.

Test misclassification cost-sensitive learning framework is extended from classic cost sensitive learning framework.

Property 3.1 *Classic cost sensitive learning is a special case of test-misclassification cost-sensitive learning model*

Proof: set all the attributes' test costs to zero, then

$$L'(x,i) = \sum_{i} p(j \mid y)C(i,j) + \sum_{k=1}^{m} p(k) * t_k$$

$$= \sum_{i} p(j \mid y)C(i,j) + \sum_{k=1}^{m} p(k) * 0$$

$$= \sum_{i} p(j \mid y)C(i,j)$$

$$= L(x,i)$$

▼

In other words, the task of test-misclassification cost-sensitive learning framework is the same as the classic cost sensitive learning when all the test costs are set as zero. Classic cost sensitive learning framework is a special case of test-misclassification cost-sensitive learning framework.

3.2.3 Test cost sensitive learning framework with multiple cost scales

Here, we extend the test cost sensitive learning from the single cost scale to multiple cost scales. We have two kinds of costs here: test and the misclassification. The simplest case is that each of them is with a single different cost scale, such as dollar cost for test and diagnosis risk for misclassification cost. In this case, test cost is not related to misclassification cost at all. But we also are appreciated with the way of incorporating the test cost into general misclassification cost decision making. So we extend the definition as follows:

Definition 3.1: In cost sensitive learning, the cost scale of misclassification cost is called *target cost scale.* All the tests are allowed to have multiple costs in different scales, all of the scales are called *resource cost scale* except that the scale is same as *target cost scale.* Misclassification costs and test cost in target cost scale are called *target misclassification costs* and *target test costs* relatively. Costs in resource cost scales are called *resource costs.*

Usually, misclassification cost is with only one cost. If there are multiple cost scales in misclassification cost, only one of them is specified as *target cost scale*, the rest scales are also called *resource cost scale* as well.

With the definition 3.1, we could still use current test sensitive learning framework to minimize the sum of misclassification cost and test cost in *target cost scale.* For costs in resource cost scales, it is not realistic to keep them minimal while minimizing the sum of target costs. For example, assume monetary is the target scale and time is the resource scale; choosing test A or test B could make a final decision. Test A need two days delay and could reduce \$200 target cost; Test B needs one day delay and could reduce \$100 target cost. You can only choose A if you want more cost reduction, or choose B if you can not bear two days delay. It is a tradeoff. Our idea is to minimize target cost and control resource costs in specific budget. So we define a "resource budget" concept for this problem as follows.

Definition 3.2: Each cost scale will be assigned a specified positive constant value with this scale, called *resource budget* of this scale.

Example 3.1: In the application of medical diagnosis, the misclassification cost is defined in scale of money, noted as "dollar". The medical tests have 3 kinds of cost in different scales: "dollar" scale for test fee, "milliliter" for blood test and "minute" for test time consuming. Then the "dollar" scale will be the *target cost scale*, and "milliliter" and "minute" will be the *resource cost scales*. We could assign the 200 milliliters and 100 minutes as their *resource budget* relatively.

The Task of cost sensitive learning with multiple cost scales is to minimize the overall cost in target cost scale and control the resource costs in specified resource budgets.

For the *Example 3.1*, the task could be to "minimize the money spent on diagnosis with maximum blood consuming 200 milliliters and finish all test needed in 100 minutes"

Following formula 3.2, assuming there are n tests, and to obtain the value of each test, we need to spend m kinds of resource costs in different cost scales $\{s_i|i=1,\ldots,u\}$. The optimal prediction for an example x in test cost sensitive learning is the class i that minimizes formula 3.2 and satisfies the constrains in formula 3.3 for each resource cost scale s,

$$\sum_{k=1}^{m} p(k) * t_k(S_i) < B(S_i) \qquad (i=1, \ldots, u) \qquad (3.3)$$

Where $t_k(s)$ is the test cost in resource scale s; $p(k)$ is the probability of performing a test for the value attribute k while making the decision; $B(s_i)$ is the resource budget for cost scale s and $B(S_i) \geq 0$.

Based on the definitions above, cost sensitive learning with single scale is a special case of the cost sensitive learning with multiple cost scales

Property 3.2 Test cost *sensitive learning framework with single cost scale is a special case of the Cost sensitive learning framework with multiple cost scales*

Proof: set all the attributes' test costs in resource to zero, then

$$\sum_{k=1}^{n} p(k) * t_k(s_i) = 0 < B(s_i) \qquad (i=1, \ldots, u)$$

So constraint 3.3 is always satisfied for each resource cost scale s. So we only need to consider formula 3.3 in target cost scale. This is just the test cost sensitive learning framework with single cost scale in section 3.2.2.

▼

With the property 3.1 and 3.2, the correlation among the three learning frameworks is shown in Figure 3.1. Our work could be useful for many diagnostic tasks involving target cost minimization and resource consumption for obtaining unknown information.

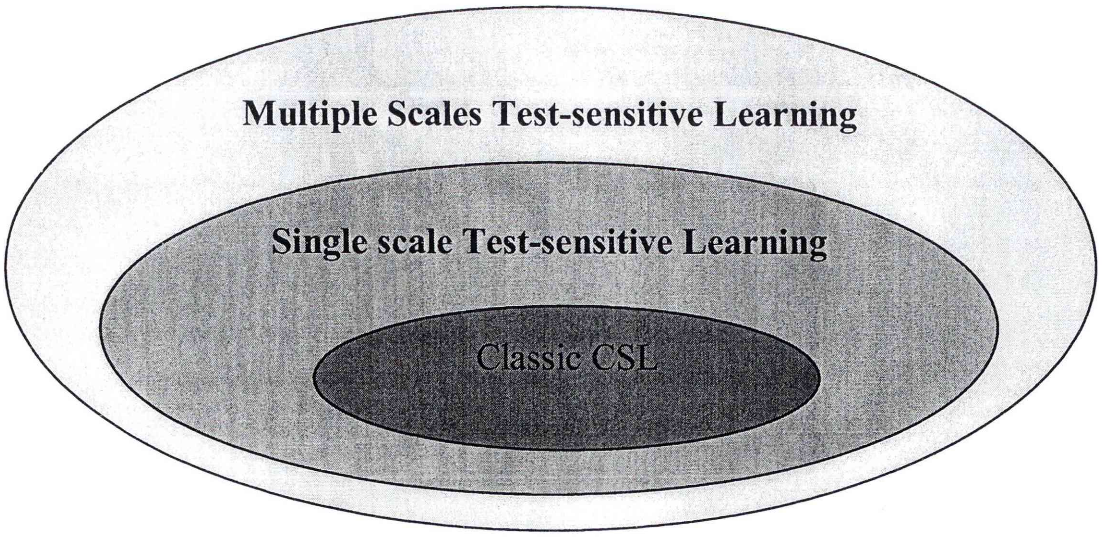

Multiple Scales Test-sensitive Learning
Single scale Test-sensitive Learning
Classic CSL

Figure 3.1 Relationship of cost-sensitive learning models

3.3 Test Cost Sensitive Learning Decision Tree with Multiple Costs Scales

In this section, we use the proposed multiple scale framework to extend the single scale test cost sensitive learning decision model in section 2.3 to a multiple scale test cost sensitive learning decision model. For easy description, we just study the case of only 1 target cost scale and 1 resource cost scale in this research. Table 3.1 shows an extended sample of Table 2.3 by involving two cost scales.

	FP	FN	A1	A2	A3	A4	A5	A6
Target (C')	800	600	50	50	50	50	50	20
Resource (C'')	0	0	20	10	10	10	10	10

Table 3.1 Test and misclassification costs with two scales set for Ecoli dataset.

There are two kinds of cost scales in Table 3.1, target cost scale (noted as C') and resource consumption scale (noted as C''). For example, we could assume the target cost is the time spent (in scale of seconds, C' = second) and resource cost is the fee spent (in scale of dollars, C''= dollar). Then we need to spend 50 seconds and 20 dollars to get the value of attribute A1. And for the misclassification cost matrix, FP1 = 800 means the patient will be charged extra misclassification target cost (800 seconds) if he is misjudge as "positive", i.e. cost of false positive.

The resource misclassification costs are usually set to zero. However, they could be non-zero values. It means we still need to resource costs occur for misclassification. We need to control the resource costs of misclassification in related resource budget. For instance, for a patient in medical domain, surgery operation is the best solution for his illness. However, the patient's physical condition is not good enough to support the surgery operation. Then we could only choose a so-called "passive" solution, such as a three-week medication treatment. The choosing of "passive solution" is a quite important and complicated problem. We will put it into future work.

Next, we will discuss the control issues, i.e. how to utilize the limited resource budget.

3.3.1 Leaf marking criteria

Decision tree uses a heuristic local optimization search approach at each candidate node. Now let's see the leaf marking criteria in section 2.3. It is actually a modified version of entropy; we called it test sensitive entropy. A class label is selected as it could minimize the formula 3.1 $L(x,i) = \sum_{j=1}^{n} p(j \mid y)C(i,j)$. The cost sensitive information entropy is T:

$$T = Min_{\text{each class } i} \{L(h(x,i)\} = Min\{\textstyle\sum_i p(j \mid y)C(i,j)\} \qquad (3.4)$$

Example 3.1: In a candidate node of a binary decision problem, assume there is a set of *P* positive and *N* negative examples respectively to be further classified by possibly building a sub-tree. And as there is no sub-tree built, we label the node as a leaf. We calculate the cost entropy as follows:

$$L(x, P) = N\times 0 + N \times FP = N\times FP$$

$$L(x, N) = P\times FN + N \times 0 = P\times FN$$

$$T = Min\{L(x, P), L(x, N)\}$$

$$= Min\{P\times FN, P\times FN\}$$

We mark the leaf as positive if P×FN > N×FP, otherwise it would be labeled as positive.

3.3.2 Attribute selecting criteria for internal nodes

Before we decide if a candidate node is a leaf, we need to check all the candidate attributes for further splitting.

There are lots of potential selecting strategies and all of them with different efficiency in cost reduction with limited resource. Here we only introduce two simplest and most intuitive ones. Both of them are with the resource cost budget constraints. Actually, one is almost the same as the single cost scale building strategies except considering the resource constraint in testing phase. And the other one is an efficiency first tree building strategy by a "performance gain" analogous measure. Anyway, for each cost scale, the cost calculation formulas are similar as in section 2.3.

Assume the training example set in current candidate node is *Y*, attribute *A* be a candidate attribute for splitting *Y*. And *A* is a discrete random variable with range $D = \{a_1, a_2, ..., a_n\}$, the target test cost of A is t_A and resource test cost is t'_A. A resource budget *B* is specified prior tree building. T is minimal target misclassification cost calculated with formula (3.4).

(1) Target-first strategy

The first one is called target-first strategy. It comes from the social point of view: target cost is invaluable. It totally ignores the resource issue and attempts to minimize the total target cost on misclassification cost and test cost. Actually we could image this case as we have unlimited resource or set all test costs as zero. It is actually the same selecting criteria as in section 2.3, because only the target cost scale used in tree building phase. Before we decide if a candidate node is a leaf, we need to check all the candidate attributes for further splitting. For target cost scale, the target cost reduction for choosing *A* to split the node is defined as:

$$G_A = \mathrm{T} - \mathrm{T(A)} = T - \sum_{a \in D} p(x)(T(Y \mid A = a) + t_A) \qquad (3.5)$$

The node will be marked as a leaf if all the candidates' cost reduction values are less then 0, otherwise the attribute with maximum cost reduction will be selected to split the node.

This criterion totally ignores resource cost in tree building phase. It means we may spend 100 resource cost to decrease 110 target cost rather than spend 50 resource cost to decrease 100 target cost. It only considers the resource budget at testing phase. Given a test example, we explore the tree and perform all needed test along the tree. Once resource budget is exhausted, we stop at current node and give a result.

(2) Performance-first strategy based on resource budget

This criterion is exactly the idea of trade-off between target and resource. It uses the target/resource gain ratio based on resource budget to choose potential splitting attributes. Assume the cost reduction of attribute *A* on target cost scale are noted G_A. All the misclassification costs on resource cost scale is zero because all the entries in misclassification cost matrix on resource cost scale are zero. In other words, if there are no missing values in attribute *A*, resource consumption of attribute A is actually the test resource cost, shown as formula (3.6).

$$R_A = \sum_{a \in D} p(x) * t'_A) = t'_A \qquad (3.6)$$

To combine the resource budget B' in tree building, we define the ***remaining resource*** ***B'*** as resource budget minus all the parent nodes' test resource cost. Attribute A will not be selected as an internal node if $B' - R_A < 0$, because *remaining resource* cannot afford to obtain the value of attribute A. In case of $B' - R_A \geq 0$, we define the ***performance gain*** of choosing A as a splitting attribute as follows:

$$PerfG_A = \frac{G_A^{\ 2}}{W * R_A(1 - B') + 1} \tag{3.7}$$

Where, w is a weight specified by domain expert. The attribute with maximum performance gain will be selected as internal node. We can see that the ratio of target cost gain and resource consuming is used to select proper attribute. And the latter part will be zero as B=100% and formula (3.7) is equal to the target first selecting strategy in formula (3.6). I.e. we will focus on target cost when we get enough resource.

Since performance-first strategy involves resource costs and resource budget during decision tree building, it is expected to outperform the target-first strategy when the resource budget is limited. The most efficient test could have more chance to be selected. At the same time, the specified resource budget B is considered in tree building. The specified budget could control both of the decision tree size and the test selections. The more resource is provided, the more tests could be selected, the more tests could be performed, and the expensive but effective tests could have more chance to be selected. We expect the performance-first strategy can provide a better overall performance than the target-first strategy.

3.3.3 Resource Control Issues

This is the first study attempting to build decision tree with multiple cost scales. An important issue is the resource controlling in building tree and predicting new examples. Our aim is to predict the class of the testing examples with the minimal total target cost, and control resource cost within a specific budget.

Because the resource costs of misclassification are all set to zero, the resource control is relatively simple. It means we could put all the resource on tests, i.e. acquiring attribute values. In testing phase, we use the similar exploring procedure as in single scale decision tree. For each new example e_i to be classified, a resource budget B_i is assigned. Of course, you could use a uniform budget for all the new examples. During the test phase, we must control the resource consumption less than the specific budget, noted by B.

Let's recall the building criteria of a decision tree. We choose a splitting attribute because it could reduce the overall target cost. It means that we could always reduce overall target cost if we use depth-first strategy to explore on a cost sensitive decision tree from root to a leaf. In other words, we should try to perform more needed tests as we can. However, with involvement of cost budget, we often fail to perform all the tests needed. Once resource is exhausted, we will stay at an internal node and stop exploring further sub-tree immediately. Then we output a class according to the target cost information of the internal node. The class label for an internal node is called ***potential class label***. As mentioned above, ***potential class label*** is reserved in each internal node, as marked in a leaf node.

Definition 3.3: For an internal decision tree node, a ***potential label*** is its class label if the internal node is labeled as a leaf. And relative target and resource misclassification cost for marking the node as a ***potential label*** is called ***potential label misclassification cost.***

Another issue is the information about the internal node when we reach the resource budget *B*. We could output the class *C* with current resource and need x extra resource to perform further test T'. And we could output the possible *potential class label* after performing test T'. Firstly, we introduce two concepts: *confirmed node* and *proposed node.*

Definition 3.4: Given a test example *S* and a resource budget *B*, exploring the decision tree from root node, when we reach an internal node *N* with the total resource consumption *R(N)* less than budget *B*, attribute value in node *N* is known but no more resource performing test for the value, then node N is called ***confirmed node,*** and each child of *N* is a ***proposed node.***

The former tells users current best decision with resource B, while the latter tells users the possible further decisions. We stop exploring the decision tree once there is not enough resource to support further exploration. Current reached node is a confirmed node and we output a result based on confirmed node. And we output the needed resource for current test, and the potential label of all proposed nodes.

3.4 Performing Tests on Testing Examples with Resource Control

For the target first strategy, i.e. the single cost scale decision, firstly we select Optimal Sequential Test (OST) strategy that performs extra tests on the unknown values along the root node to leaf. Once resource is exhausted, we stop right there, and use the ratio of positive and negative examples in that (internal) node to predict the testing example (recall that these ratios are calculated based on training cases which also have unknown values at this node). In other words, the original tree will be "pruned" to meet the budget requirement. Three different-size trees for Ecoli data in Figure 3.2 show the testing procedure: tree (a) is the original tree with enough resource; tree (b) is pruned from tree (a) as resource budget 50 which could support the test of A6; tree (3) directly gives a decision as no test could be performed with resource budget 20.

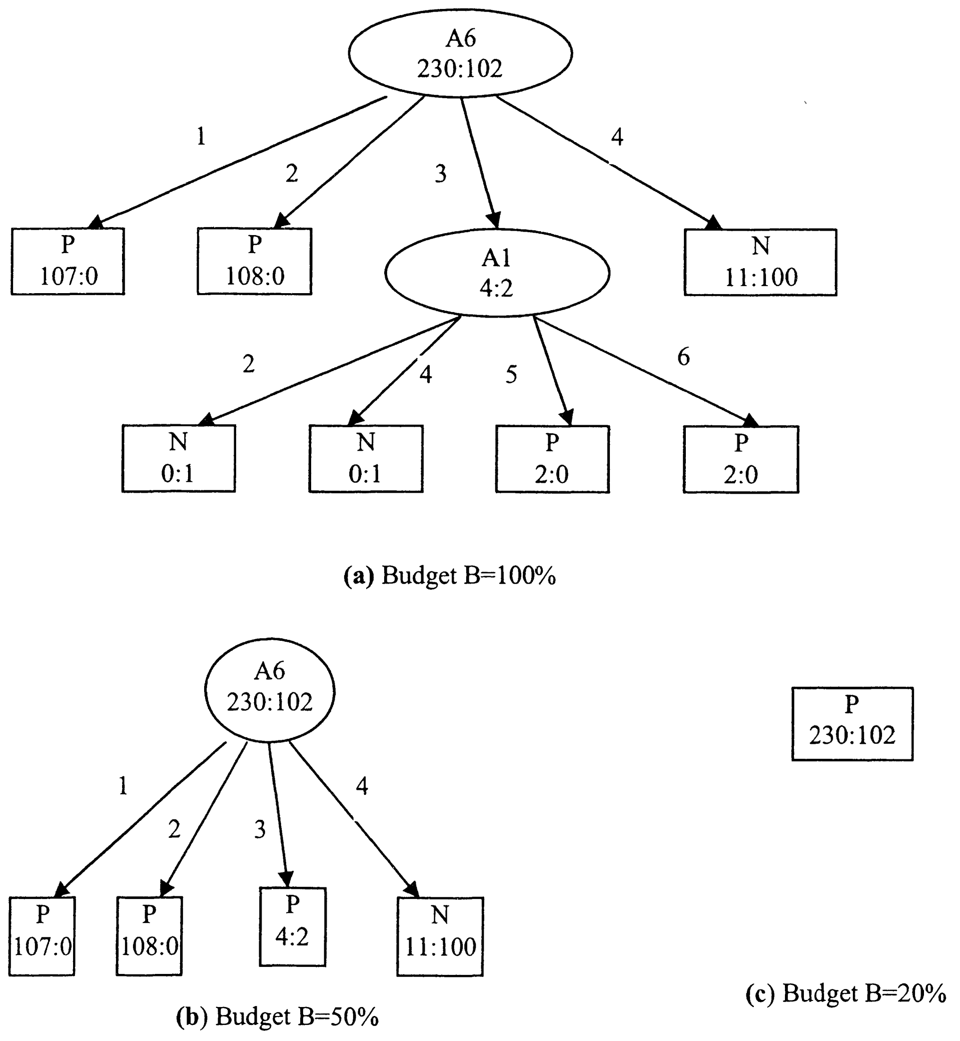
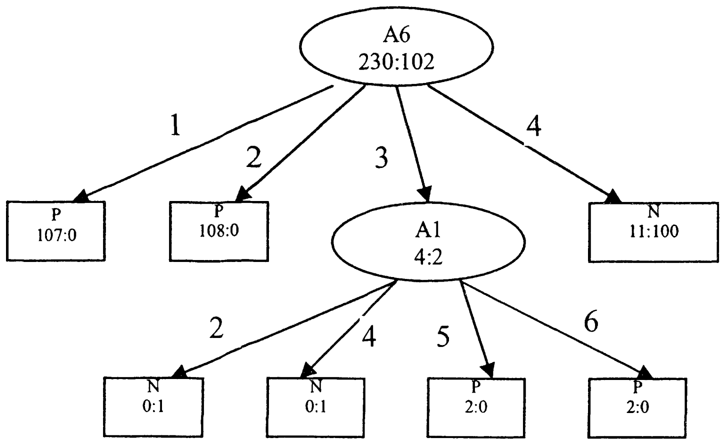
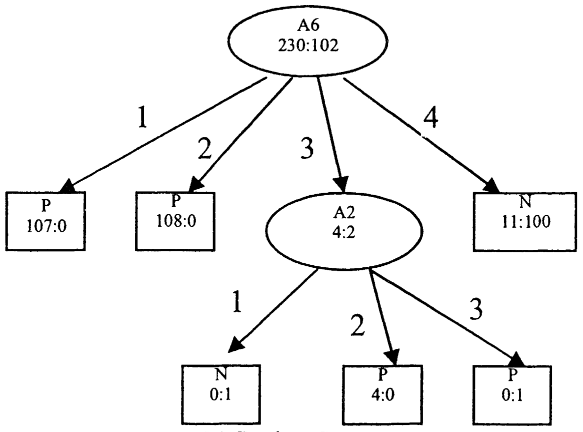


Figure 3.2 Three different decision trees for Ecoli data (single cost scale) built with different resource budgets.

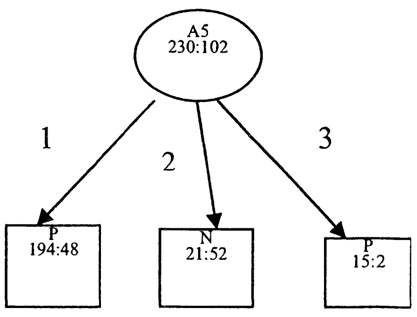
For the performance first strategy, we may get different decision trees with different resource budgets because the resource and budget are considered in the tree building phase. Each tree selects proper tests with given resource budget. Three trees are built and shown in Figure 3.3 with the resource budget setting 100, 50 and 20. Following figures show variant decision trees built from Ecoli dataset with varied resource budgets, misclassification cost, test cost and resource cost are set as per Table 3.1.



(a) Budget B=100%



(b) Budget B=50%



(c) Budget B=20%

Figure 3.3. Three different decision trees for Ecoli data (multiple cost scales) built with different resource budget

Figure 3.4 shows the average target costs for the Ecoli dataset in terms of above three percentages of resource budgets 20%, 50% and 100%. According to the curves in Figure 3.4 the average target cost of performance first strategy is significantly less than target first strategy in resource budgets 20%. And the two strategies will have same average target cost in 100% resource budget. It means the strategies are the same as enough resource available.

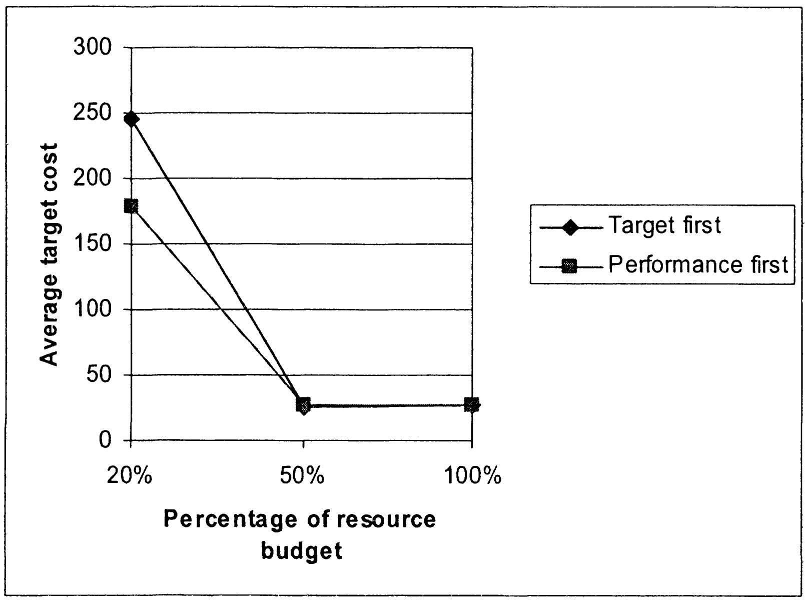


Figure 3.4 Comparing the total cost under 3 different resource budgets

3.5. Experiments

The requirements of decision tree are rather low and running time is not the key concern of this research. We conduct the experiments on a PC with P4 2.4Ghz cpu, 512M memory, windows 2000 system. All the programs are extended from original Java codes of machine learning and data mining software package Weka [WF05], running on the JAVA 1.5.

We conducted experiments on four real-world datasets [BM98] and compared the target-first and performance-first tree building strategies against C4.5. These datasets are chosen because they have at least some discrete attributes, binary class, and a good number of examples. The numerical attributes in datasets are discretized first using minimal entropy method [FI93] as our algorithm can currently only deal with discrete attributes. The datasets are listed in Table 3.2.

	No. of attributes	No. of examples	Class distribution (P/N)
Ecoli	6	332	230/102
Breast	9	683	444/239
Heart	8	161	98/163
Australia	15	653	296/357

Table 3.2 Datasets used in the experiments.

We randomly assign random numbers between 0 and 100 to each attribute as test target costs and resource. We also assign 800 for false positive, and 600 for false negative misclassification costs. The cost of true positives and true negatives is set to 0. Each dataset is then split into training and test sets using 10-fold cross validation (thus test sets also have the same percentages of missing values). To compare the influence of resource budget on two strategies, we conducted experiments on each split with varying budget B that only supports a part of all needed tests from 20 to 100 percent. For Target First Strategy, only one decision tree is built from the training dataset in each split, and resource budgets are applied to the test examples by sequential exploring the tree till resource is exhausted. For Performance First Strategy, in each split, trees might be different for different resource budgets, but we could always reach the leaf with related budgets.

Figures 3.5 and 3.6 show detailed results of average target costs and resource utilizations (shown in percentage of the total resource cost) for the Ecoli dataset with above setting, whereas results on the other datasets are similar and are only listed in Table 3.3 and 3.4. The scales on the x-axis (20%, 40%, and so on) represent the percentage of resource budget. The curves in Figure 3.5 and Figure 3.6 represent the difference in average target cost and the average resource of two different strategies.

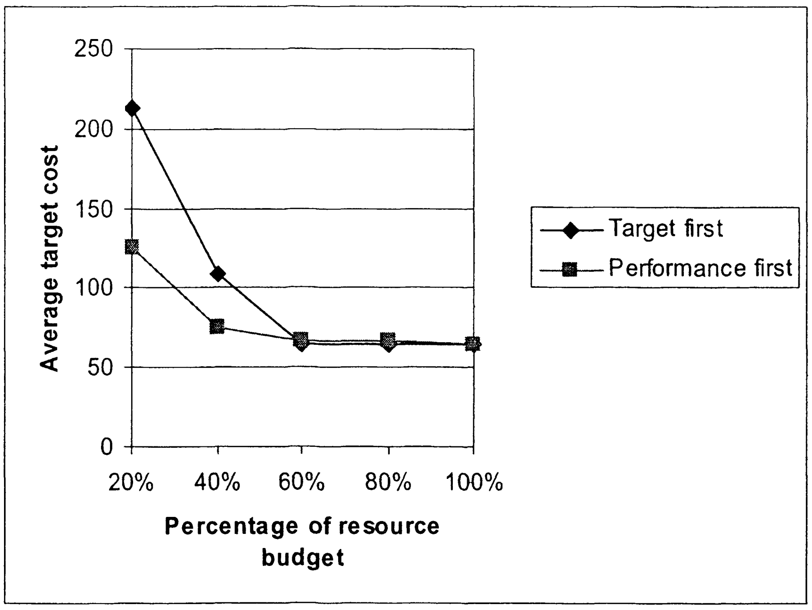


Figure 3.5. Comparing the total cost under different resource budgets

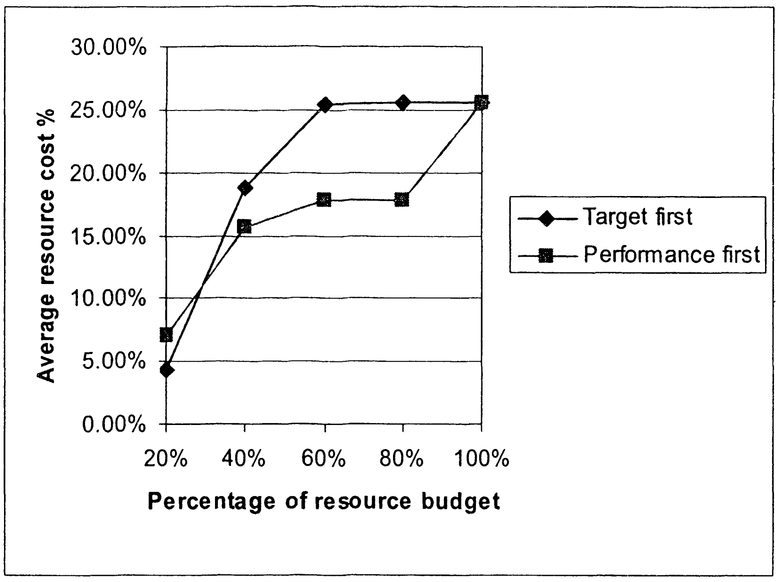


Figure 3.6. Comparing the resource utilization (percentage) under different resource budgets

The experimental results show that when resource budgets are less than 50%, the Performance first strategy outperforms the Target first strategy. It reduces the total of misclassification cost and the test cost by maximizing the utilization of the limited resources. When resource budgets are more than 50%, the performance of the two strategies are very close. If the resource budget reaches 100%, the two strategies build the same decision tree and the misclassification cost, test cost and resource cost are exactly the same. This means that with enough resource budgets, the Performance first strategy and the Target first strategy tend to build very similar decision trees and get similar results. However, when resource budgets decrease, the Performance first strategy tends to select attributes that with less resource costs to split, and build decision trees that use the limited resources more efficiently. This reduces the total of misclassification cost and test cost in the end.

Results on the other datasets are listed in Tables 3.3 and 3.4. Those results are quite similar to the results in Ecoli dataset. Table 3.3 is about average target costs and Table 3.4 is about average resource utilization (percentage of the total resource cost).

	20%	40%	60%	80%	100%
The Ecoli Dataset					
Target First Strategy	212.7	108.4	64	64	64
Performance First Strategy	125.4	75.6	66	66	64
The Breast Dataset					
Target First Strategy	79.1	83.7	85.6	85.6	85.6
Performance First Strategy	78.7	81.9	82.9	82.9	85.6
The Heart Dataset					
Target First Strategy	209.1	227.3	227	227	227
Performance First Strategy	206.1	225.8	226.2	227.6	227
The Australia Dataset					
Target First Strategy	148.5	148.5	148.5	148.5	148.5
Performance First Strategy	148.5	148.5	148.5	148.5	148.5

Table 3.3 Average Target Cost with Two Tree Building Strategies under Five Different Percentages of Resource Budgets in the Data Sets

	20%	40%	60%	80%	100%
The Ecoli Dataset					
Target First Strategy	4.%	9%	25%	26%	26%
Performance First Strategy	7%	16%	18%	18%	26%
The Breast Dataset					
Target First Strategy	11%	12.6%	12.8%	12.8%	12.8%
Performance First Strategy	10%	11%	12%	12%	12.8%
The Heart Dataset					
Target First Strategy	10%	12.5%	12.6%	12.6%	12.6%
Performance First Strategy	10%	12%	12.6%	12.6%	12.6%
The Australia Dataset					
Target First Strategy	6%	6%	6%	6%	6%
Performance First Strategy	6%	6%	6%	6%	6%

Table 3.4 Average resource utilization with Two Tree Building Strategies under Five Different Percentages of Resource Budgets in the Data Sets

3.6 Conclusions and Future Work

In this chapter, we have presented a simple and efficient framework to overcome the difficulty in defining the multiple costs on the same cost scale in cost sensitive learning. Then we apply the proposed framework test cost sensitive decision tree involving two kinds of cost scales; minimizing the one kind of cost and controlling the other one in a given budget.

We proposed a performance-first splitting criterion for attribute selection, and discussed several intelligent testing strategies in single cost scales as involving resource control. Our experiments show that budget-based splitting criterion dramatically outperforms the target-first tree building strategy from single scale decision tree. In addition, compared to other related works, our algorithm has lower total cost in most cases, and is thus more robust and practical.

In the future, we plan to consider how to minimize the total target cost with partial cost-resource exchanging. In some situations, such as medical diagnosis, this scenario is more practical since some hospitals provide VIP services. We also want to extend our Optimal Sequential Test to Optimal Batch Test, and pruning can also be introduced in our tree-building algorithm to avoid over-fitting of the data.

Chapter 4 Utilization based Test Cost Sensitive Decision Trees

As shown in section 2.3, unlabeled examples may contain partial known information before being given a prediction. However, the decision tree is built based on the assumption that their values are unknown before test. Obviously, it is not rational for this situation. Firstly, we don't need to perform extra tests for such information. Secondly, known information may influence the attribute selection in decision tree building phase. In this chapter, we study how to utilize the partial known information in test cost sensitive learning. In section 4.1, a revised lazy tree algorithm is proposed for the above problem. For each new patient, we first introduce a lazy test-sensitive decision tree based on known values, which utilizes known information and saves test cost for the patient. Then we extend the tree by considering the attributes with unknown values. It reduces overall cost by avoiding redundant tests. In Section 4.2, we discuss the batch-tests problem for the test cost sensitive decision trees. For an unlabeled example with some unknown attribute values, multiple tests are chosen and done in one shot, rather than in a sequential manner in the test-sensitive tree. A hybrid batch test strategy with known information utilization and global resource control is proposed to select proper batch tests and keep the minimization of overall target cost. .

4.1 Lazy Test Cost Sensitive Decision Trees with Multiple Cost Scales

4.1.1 Motivation

As shown in section 2.3, the single cost scale decision tree algorithm assumes that unlabeled examples may contain partial known information before being given a prediction. We are interested in this assumption because it is closely related to the real world medical diagnosis application. Some information could be found in the patient's history records.

However, the single cost scale decision tree algorithm builds an overall tree based on the assumption that their values are all unknown. Obviously, it is not rational for this situation. Firstly, the attributes with known values should not be assigned a test cost. Secondly, known information may influence the attribute selection in decision tree building phase. In this chapter, we study the ways to utilize known information. A novel yet efficient approach is proposed to deal with this problem based on the test sensitive decision tree with two cost scales. Firstly, we revise the test cost table according to known values. Secondly, a lazy test-sensitive decision tree for each unlabelled instance based on its resource budget. Then we compare our algorithm with the current test sensitive decision tree algorithm with single cost scale. Here we give a briefly introduction of classic lazy decision tree.

4.1.2 Classic lazy decision tree

Lazy Decision Tree is a lazy algorithm for inducing decision tree [FYK96], called LazyDT. LazyDT doesn't build a general tree for all unlabelled instances that are waiting to be assigned a class label, but build an individual tree for each instance according to instance's information.

In Figure 4.1 we briefly describe the general steps of the LazyDT algorithm. The core part of the algorithm is how to select a test. LazyDT chooses a test that optimizes the resulting branch taken by the given test instance. Once a test is selected, only instances that take the same branch as the test instance are kept to build the remaining part of the lazy decision tree. We omit the details of the algorithm and refer readers to the original paper [FYK96] for an exact description of the LazyDT algorithm.

```
Inputs: S is the training set
        y is the test instance to be classified
Output: class label for the test instance y
1. If all instances in S are from a single class l, return l.
2. Otherwise, select a test T and let t be the value of the test on instance y.
   Let $S_0$ be the set of training instances satisfying T = t and apply the
   algorithm to $S_0$ and $y$.
```

Figure 4.1 A generic lazy decision tree algorithm

LazyDT has some merits in comparison with regular decision tree algorithms [FYK96]. First, the decision paths built by LazyDT are often shorter and therefore more comprehensible than paths of regular decision trees. Second, it is well known that given limited training data, regular decision tree algorithms may suffer from the data fragmentation problem [PH90]. Regular decision tree algorithms select a test for the root of each sub-tree based on the average improvement of the test selection criterion (such as entropy). Because the choice is based on average improvement, a particular child branch of the test may see a decrease in the value of the criterion or remain the same. In instances of taking such a branch, the test may be detrimental since it fragments the data unnecessarily. This may causes the resulting path to be less accurate because the remaining tests are selected based on fewer training instances. In contrast, LazyDT constructs a customized "tree" for each test instance, which consists of only a single path from the root to a labeled leaf node. Given a test instance, LazyDT selects a test by focusing on the branch that will be taken by the test instance. By doing so it avoids unnecessary data fragmentation and may produce a more accurate classifier for the specific instance.

Given the above strengths of LazyDT, we are interested in further applying it to build a dynamic cost sensitive tree for a test instance with partial known information. However, some necessary revision is needed because the domain and problem definition are different. In our problem, the unknown values in test instances could be obtained after spending a target test cost and a resource cost, where, classic LazyDT just ignores the attributes with unknown values during tree building phase. On the other hand, we still need to consider the resource control issues as well. In the following section, we will discuss two ways to utilize the known information and reduce overall cost.

4.1.3 Lazy Test Cost Sensitive Decision Tree with Two Cost Scales

As mentioned in section 4.1, a new example may contain known information before being classified. The tests of those known attributes should be free. We must reassign the costs of the known attributes to be \$0 while the cost of the unknown attributes remains unchanged. According to the results in single scale test sensitive decision tree [LSY06], this small change could reduce the total misclassification and test cost significantly.

Example 4.1: Assume we have a test example in Ecoli dataset as in Table 2.4 to be predicted. Attribute A2, A4, A5 are known before prediction. Its original test costs and misclassification cost with two cost scales are set as in Table 3.1. Then Table 3.1 will be modified as Table 4.1 before we build a lazy decision tree.

	FP	FN	A1	A2	A3	A4	A5	A6
Target (C')	800	600	50	0	50	0	0	20
Resource (C'')	0	0	20	0	10	0	0	10

Table 4.1 Test and misclassification costs with two scales set for an example of Ecoli dataset with some known values

Lazy tree can utilize information in the known attributes and reduce redundant tests. In Figure 4.2, an algorithm of lazy test cost sensitive decision tree with two cost scales learning based on the multiple cost scale framework in section 3.2 is presented.

Algorithm 4.1 Lazy-csDT (D; A;CL;R; C_T; C_R;AS; LM; y)

Inputs:

D —a data set of samples $\{x_1; x_2; \ldots ; x_n\}$,

A —a set of attributes $\{A_1;A_2; \ldots ;A_m\}$,

CL —predefined classes $\{c_1; c_2; \ldots ; c_p \}$,

R —misclassification cost matrix,

C_T—a test cost vector in target scale,

C_R—a test cost vector in resource scale,

AS (D, A, CL, **R, C_T, C_R**,)—attribute selection formula for internal nodes, the result is an attribute,

LM (D, A, CL, **R, C_T, C_R**)—leaf marking formula, the result is a class,

y — is the value vector of test instance to be classified, unknown attribute is marked as null,

Output: class label for the test instance y

```
1. If there is no attribute satisfying the AS (D, A, CL, R, $C_T$, $C_R$,), return LM (D, A, CL, R, $C_T$, $C_R$),

2. Otherwise, select attribute $A_i$ = AS (D, A, CL, R, $C_T$, $C_R$,), and let t be the value of the attribute on instance y;

    Let $D_0$ be the set of training instances satisfying T = t and recursively apply the
      Algorithm Lazy-csDT on $D_0$
```

Figure 4.2 Lazy test sensitive decision tree algorithm with two cost scales

We adopt the same leaf marking formula LM as in section 3.3.1. And we will compare the performance of the multiple scales lazy tree model with the single scale lazy tree model in section 2.3. Multiple scales lazy tree uses *Performance-first* strategies and Multiple scales lazy tree uses *Target-first* in attribute selection of each internal node (the two strategies are defined in Section 3.3.2). Actually, lazy tree only produces a path along the root node to a leaf. Single scale lazy tree model only builds one lazy tree for different resource budgets. To control the resource in a specific budget, we explore the path step by step. Once the resource is exhausted, stop exploring further sub-tree and output a class according to the target cost information of the current node. Multiple scales lazy tree model builds different lazy tree for different resource budgets. The length of the path is controlled by the resource budget.

4.1.4 Performance Evaluation

We still conducted experiments with same datasets and cost setting as in section 3.5. For each example being predicted, we randomly choose a part of attributes and set their values as known. Then we modify the test cost table according to the known information. Variant lazy trees are built and used to predict the example's class based on different attribute selection strategies and resource budgets.

Here we give the detailed results of average target costs and resource utilizations (shown in percentage of the total resource cost) for the 4 dataset, as shown in Figure 4.3 to 4.6. The scales on the x-axis (20%, 40%, and so on) represent the percentage of resource budget. The curves in Figure 4.3 and Figure 4.6 represent the difference in average target cost and the average resource of two different strategies.

The experimental results are similar to the trends as in section 3.5, the Performance first strategy outperforms the Target first strategy. It reduces the total of misclassification cost and the test cost by maximizing the utilization of the limited resources. When resource budgets are more than 50%, the performance of the two strategies are very close. If the resource budget reaches 100%, the two strategies build the same decision tree and the misclassification cost, test cost and resource cost are exactly the same.

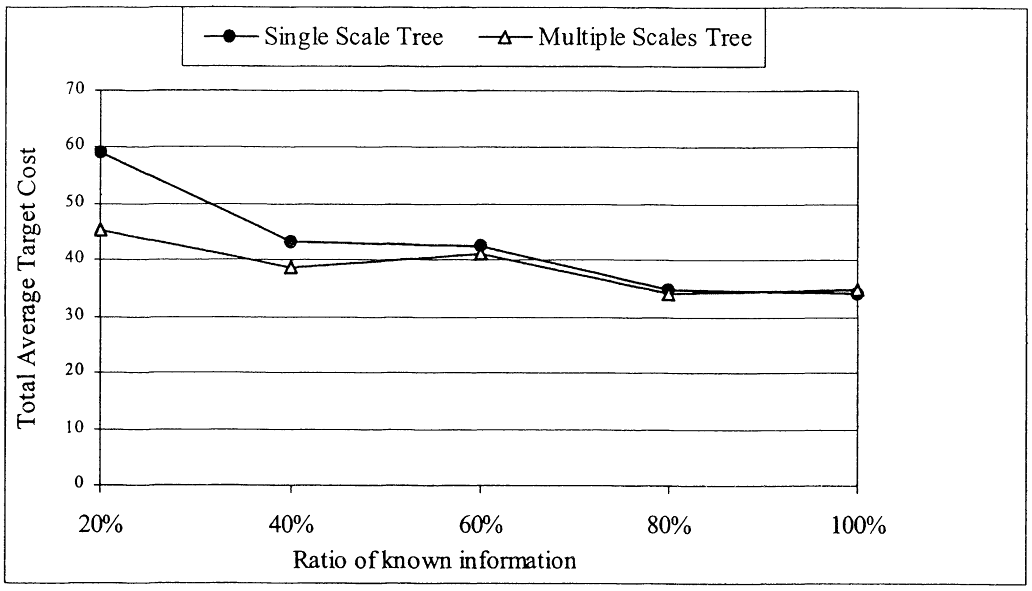


Figure 4.3. The total average target costs of single and multiple cost scales tree under different resource budgets (Dataset Ecoli).

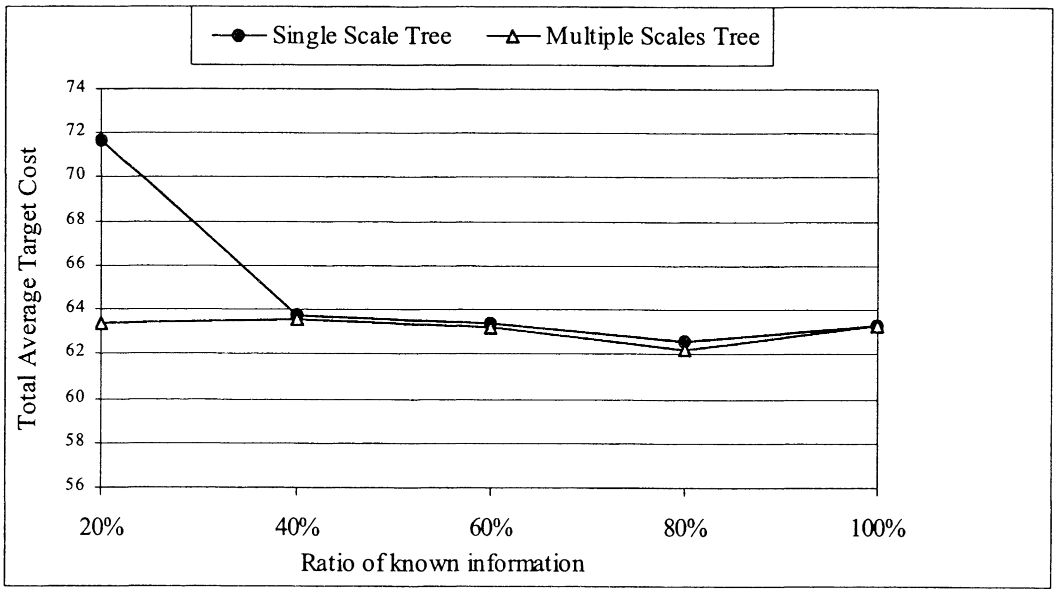


Figure 4.4. The total average target costs for Dataset Breast.

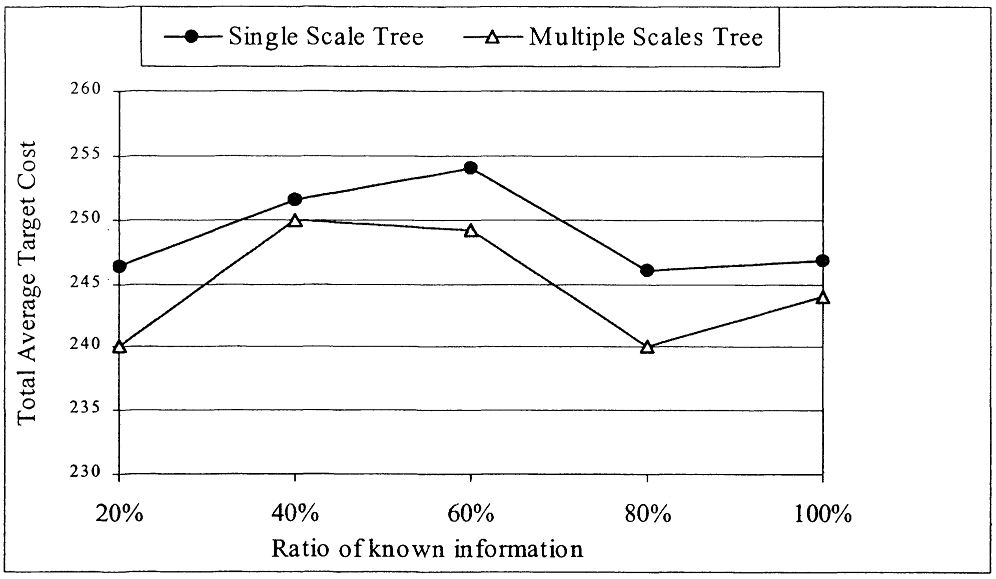


Figure 4.5. The total average target costs for Dataset Heart Disease.

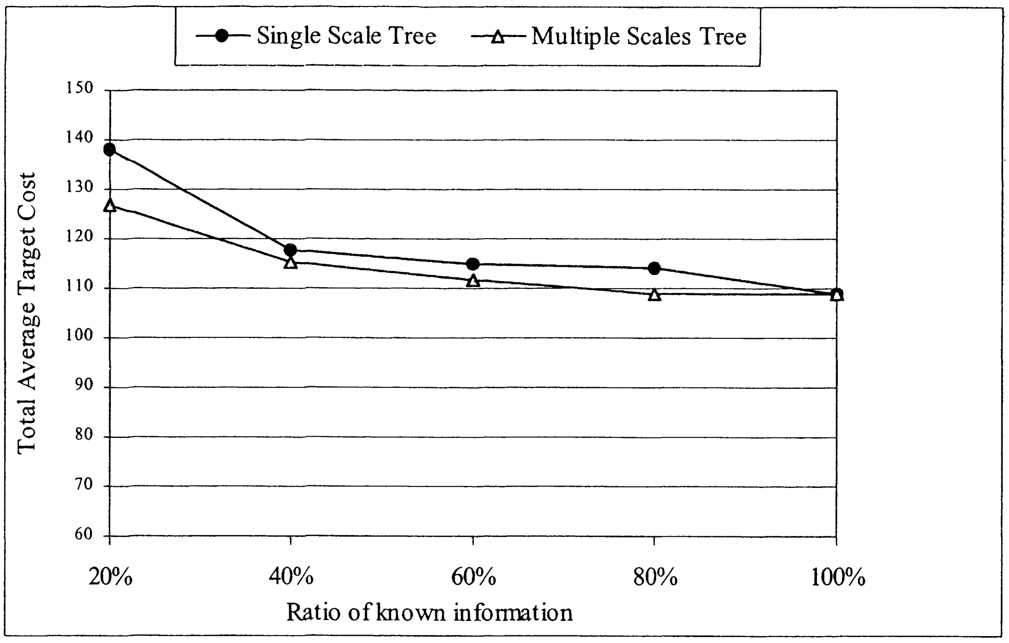


Figure 4.6. The average total target costs for Dataset Australia.

4.2 Batch Testing Strategies for Test Cost Sensitive Decision Trees

In some real world applications, such as medical diagnosis, doctors need to select sets of medical tests in sequence in order to make an accurate diagnosis of patient diseases. Very often, doctors select a batch of tests for a patient, i.e. multiple tests are chosen and done in one shot, rather than in a sequential manner in the test-sensitive tree. Based on the results of the batch of tests, doctors could also choose another batch of tests for the patient till a final diagnosis is made. In this section, we discuss the batch-tests problem for the test cost sensitive decision trees.

4.2.1 Batch Tests Selection

For an unlabeled example with some unknown attribute values, multiple tests are chosen and done in one shot, rather than in a sequential manner in the test-sensitive tree. A hybrid batch test strategy with known information utilization and global resource control is proposed to select proper batch tests and keep the minimization of overall target cost. Our work will be useful in many urgent diagnostic tasks involving target cost minimization and resource consumption for obtaining missing information. This chapter is related to my publication in [FSDK05, QZZ05, QZZ06].

Current batch tests selection methods are based on an overall tree built from training set. It puts all the attributes internal nodes into a queue by sequence of breadth-first exploring the tree. Duplicated attributes will be removed from the queue. A set of n tests to obtain the values of top n attributes in the queue is called a *batch of n tests.*

Example 4.2: Given an example of test sensitive decision tree Ecoli Dataset as in Figure 4.7. The complete queue is A5 => A6 => A3 => A2 => A4; then the *batch of 5 tests* is {A5, A6, A3, A2, A4} and the *batch of 3 tests* is {A5, A6, A3 }

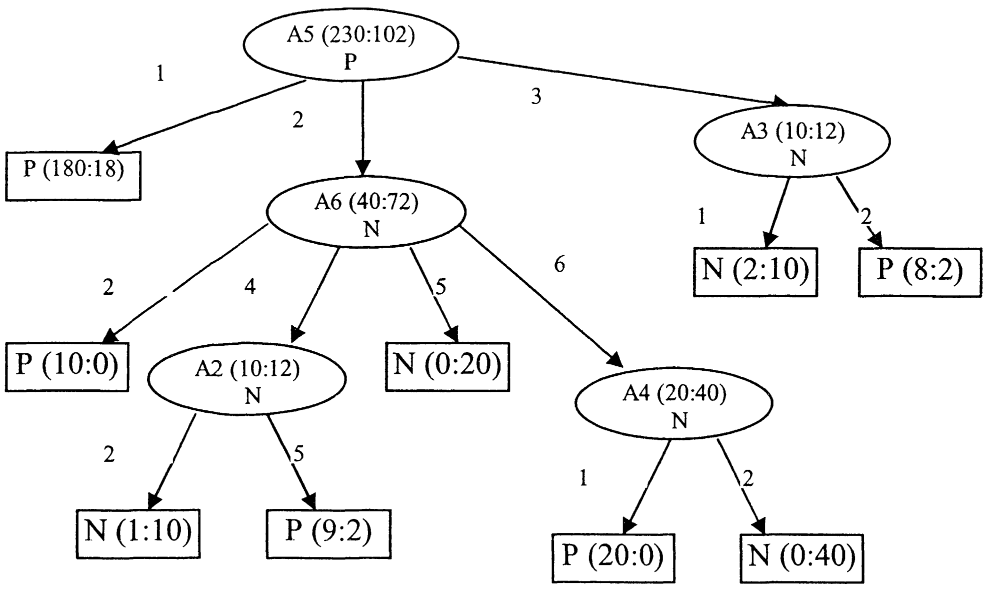


Figure 4.7. An overall test cost sensitive decision tree

To select proper batch of tests for a specific patient, doctors usually select different batch according to the patient's known information, such as historic records, direct observation, etc. For example, *A2, A4* will not be considered if *A6* is known already; *A6* even has a chance to replace attribute A5 as the root node. In the language of cost sensitive learning, a lazy tree is proper for the patient. However, current lazy tree algorithms always build a path of nodes from the root node to a leaf. It is not proper for the batch test problem. We don't know which branch to go before the whole batch tests are performed. Following the example tree in Figure *4.7*, both of the attributes A2 and A6 should be taken into account if attribute A6 is unknown. In the next section, a hybrid lazy test cost sensitive tree algorithm is proposed for batch test selection.

4.2.2 Hybrid Lazy Tree for Batch Tests Selection

The hybrid strategy is motivated from real world diagnostic procedure. For instance, when a patient goes to see a doctor, he usually tells doctor his basic symptoms and his medical history information. The simple information could be regarded as free and already known. Then doctor has a rough decision, and several possible classes could be distinguished. Patient will be asked to perform some further tests to obtain more information till the information is enough for a satisfying diagnosis. Those later tests are not free and doctor need to consider minimizing the overall target misclassification cost and test cost. Sometimes he needs to choose some economic tests according to the patient's finance situation.

Here we present a hybrid lazy tree building strategy for batch tests selection. We adopt the leaf marking and the performance-first tree strategy in Chapter 3 for decision tree building. Assume we need to classify an example with several known attributes and control the resource cost in a budget *B*. If a known attribute is chosen as an internal node, we will prune all its sub-trees except the branch corresponding to the attribute's value. The ***Hybrid-Lazy-csDT*** algorithm is shown in 4.8. It is a revised version of algorithm 4.1 **Lazy-csDT** in section 4.3.

According to the algorithm, hybrid lazy tree is the same as the general tree if all the attributes are unknown; and hybrid lazy tree is the same as lazy tree if all the attributes are unknown.

Algorithm 4.2 Hybrid-Lazy-csDT (D; A;CL;R; C_T; C_R;AS; LM; y)

Inputs:

D —a data set of samples $\{x_1; x_2; \ldots; x_n\}$,

A —a set of attributes $\{A_1; A_2; \ldots; A_m\}$, $A_m \in \{v_{m;1}; v_{m;2}; \ldots; v_{m;|Am|}\}$

CL —predefined classes $\{c_1; c_2; \ldots; c_p\}$,

R —misclassification cost matrix,

C_T —a test cost vector in target scale,

C_R —a test cost vector in resource scale,

AS (D, A, CL, **R**, **C_T**, **C_R**,)—attribute selection formula for internal nodes, the result is an attribute,

LM (D, A, CL, **R**, **C_T**, **C_R**)—leaf marking formula, the result is a class,

y — is the value vector of test instance to be classified, unknown attribute is marked as null,

Output: Modele(y)—the hybrid lazy tree learned model that predicts the class value of a new case y with a probability measure.

1. If there is no attribute satisfying the *AS (D, A, CL, **R**, **C_T**, **C_R**)*, then stop at D with a class label *C_D=LM (D, A, CL, R, CT, CR)*,

2. Otherwise, select attribute A_i = *AS (D, A, CL, **R**, **C_T**, **C_R**)*, let $D_{i,j}$ be the set of training instances satisfying $A_i = v_{i;j}$ ($j \in \{1,\ldots, |A_i|\}$)

if A_i is known and $A_i = v_{i;t}$, then only apply TCSL-learn algorithm ***Hybrid-Lazy-csDT*** to $D_{i,t}$; otherwise apply TCSL-learn algorithm ***Hybrid-Lazy-csDT*** to each $D_{i,j}$, ($j \in \{1,\ldots, |A_i|\}$)

Figure 4.8 Lazy test sensitive decision tree algorithm with two cost scales

Example 5.1: Let's see an example for test instance in Table 4.1. In Tables 4.1, values of attributes *A2, A4*, and *A5* are known. Assume attribute *A5* is selected root node. Because *A5*'s value is 2, it goes down to the second branch, shown as in Figure 4.9. In this case, we only keep the second branch and prune rest branches. Assume attribute *A6* is selected to split the second branch of root node. The value of attribute *A6* is unknown, so we need to expand all its branches. Finally, we may get a decision tree as shown in Figure 4.10.

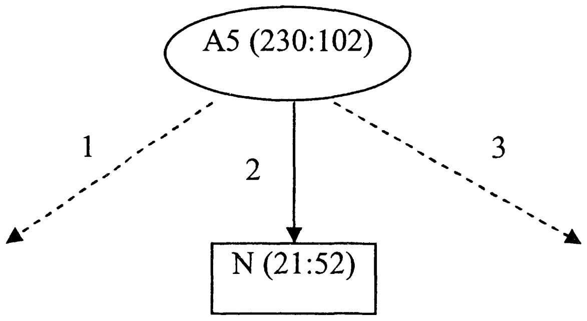


Figure 4.9. .Choosing known attribute as an internal node

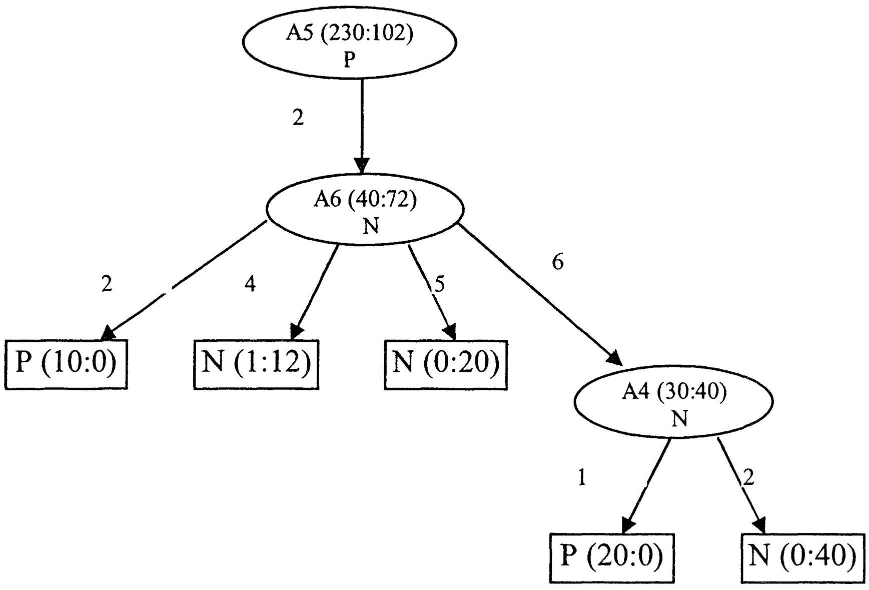


Figure 4.10 A hybrid lazy decision tree extended from figure 5.2.

4.2.3 Performance Evaluation

We compare the performance of batch selection in hybrid lazy tree with the batch selection in an overall tree built on training set. We still use the four datasets in section 3.5. We set four different known attributes ratios from 20% to 100% for batch test selection. The results on total target cost are shown in Figure 4.11 to Figure 4.14. We can see that hybrid lazy tree strategy outperforms the general tree strategy in most of cases. It is rational because hybrid lazy tree strategy prunes the redundant branches of internal nodes split by known attributes.

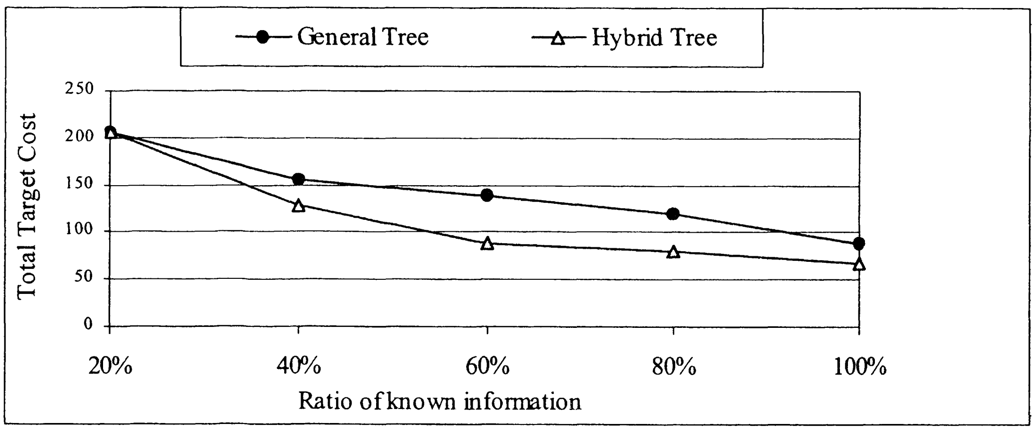


Figure 4.11. Total target costs with different ratio of known attributes on dataset Ecoli. Target costs in two strategies go down when more known attributes are available.

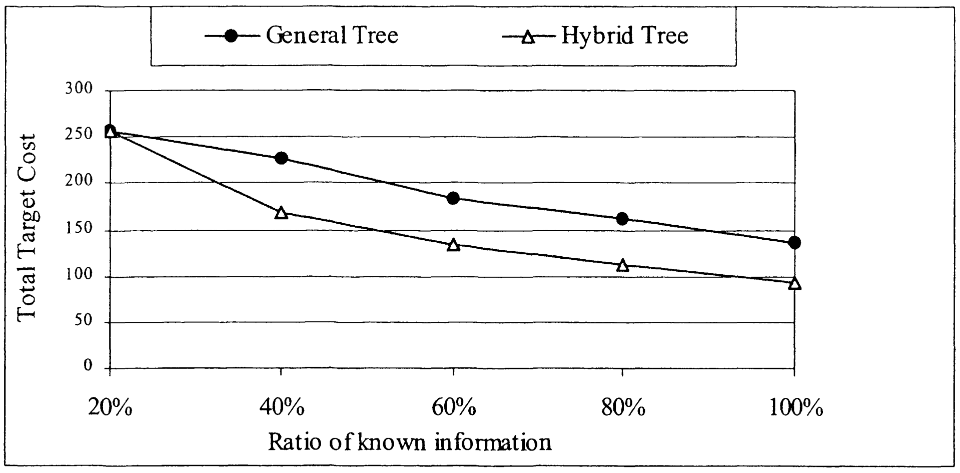


Figure 4.12. Total target costs with different ratio of known information on dataset Breast

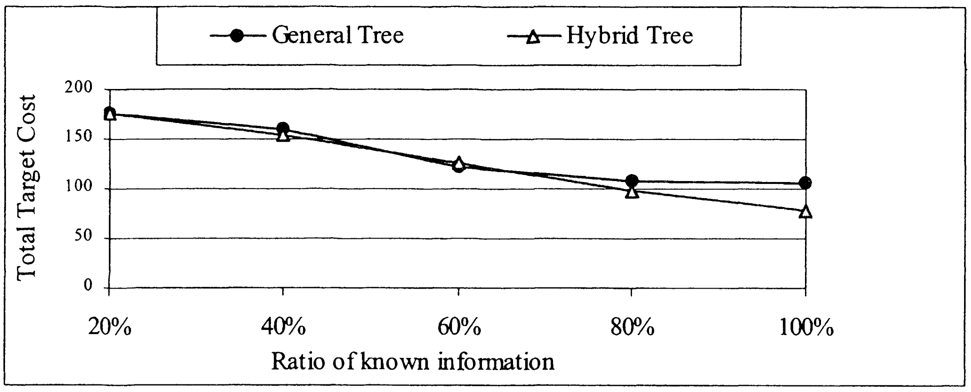


Figure 4.13. Total target costs with different ratio of known information on dataset Heart

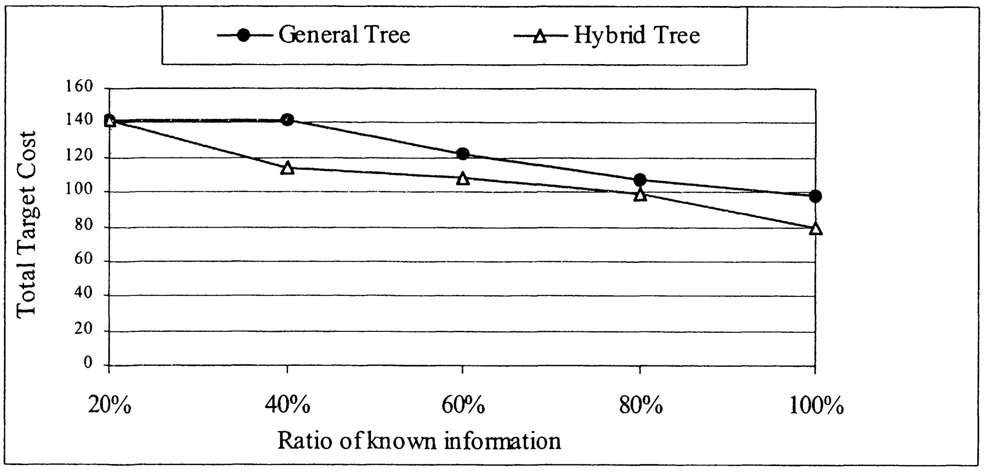


Figure 4.14. Total target costs with different ratio of known information on dataset Australia

4.3 Conclusions and Future Work

In this chapter, we discussed the situation that part of unlabelled example's information is known before performing a cost sensitive learning. We noticed it is not rational to build an overall decision tree for this situation. With the multiple cost scale learning framework, a lazy test cost sensitive learning with two cost scales is proposed to tackle the problem.

We also discussed the batch test selection problem. A hybrid lazy tree based selection strategy is proposed to utilize the known values. Our experiments show that our new batch test strategy outperforms the other general tree strategy in most of cases.

The problems addressed in this chapter are interesting and significant. They are with very strong real work background, such as medical diagnosis, insurance application evaluation, etc. In the future, we plan to perform further study on the utilization problems in test cost sensitive learning. We also plan to consider doing further research on the batch test selection. A potential way is to combine the existing work in domain of feature selection.

Chapter 5 Absent and Missing Values in Cost-Sensitive Decision Trees

Many real-world datasets for machine learning and data mining contain unknown values, and much previous research regards it as a problem, and attempts to impute unknown values before training and testing. In this chapter, we study this issue in cost-sensitive learning that considers both test costs and misclassification costs. If some attributes (tests) are too expensive in obtaining their values, it would be more cost-effective to miss out their values, similar to skipping expensive and risky tests (missing values) in patient diagnosis (classification). In other words, some data fields may be left as blank intentionally, noted as "absent data". From our point of view, "absent is useful" as absent values actually reduce the total cost of tests and misclassifications, and therefore, it is not meaningful to impute their values. In this chapter, we first discuss and compare several strategies that utilize only known values and that "missing is useful" for cost reduction in cost-sensitive decision tree learning. Then we give a further discussion about it, especially how absent values are different from missing values and how to distinguish them?

The rest of the chapter is organized as follows. In Section 5.1, we give a basic introduction. In Section 5.2 we review previous techniques for dealing with missing values, and a recent cost-sensitive decision tree algorithm based on which we will discuss our missing-value strategies. In Section 5.3, we discuss and compare four missing-value strategies of test cost sensitive learning. In Section 5.4, we discuss how to identify the absent values from missing values with lazy tree. In Section 5.5, new definition and algorithm for patching missing data are proposed to handle the identified missing values. In Section 5.6, we experimentally study the proposed algorithms using real-world datasets. Finally, our conclusions and future work are presented in Section 5.7.

This chapter is related to my publication in IEEE TKDE [ZQLS05] and AMT06 [QZZ06].

5.1 Introduction

5.1.1 Missing fields in data set

Machine learning and data mining rely heavily on a large amount of data for building learning models and making predictions, and thus, the quality of data is ultimately important. Though there is no formal measure on the quality of data, it can be intuitively quantified by the inclusion of relevant attributes, the errors in attribute values, and the amount of missing values in datasets. This chapter studies the issue of missing attribute values in training and test datasets.

Indeed, many real-world datasets contain missing values, and it is often regarded as a difficult problem to cope with. Sometimes values are missing due to unknown reasons, or errors and omissions when data are recorded and transferred. As many statistical and learning methods cannot deal with missing values directly, examples with missing values are often deleted. However, deleting cases can result in a loss of a large amount of valuable data. Thus much previous research has focused on filling or imputing the missing values before learning and testing are applied to.

In this chapter, we study missing data in cost-sensitive learning in which both misclassification costs and test costs are considered. That is, there is a known cost associated with each attribute (variable or test) when obtaining its values. This is true in most real-world application where it costs money to obtain new information. For example, in medical diagnosis, it costs money (to the patient, lab, or health insurance) to request blood tests, X-ray, or other types of tests, some of which can be quite expensive and even risky to patient life (which can also be converted to cost). Doctors often have to balance the cost effectiveness of the tests and the accuracy of the diagnosis (prediction) to decide what tests should be performed. That is, if a test is too expensive compared to the potential reduction in misclassification cost, it is desirable to skip the test. In other words, if the goal is to minimize the total cost of tests and misclassifications, some attribute values *should* be missing, and doctors do not need to know the missing values in their diagnosis (prediction or classification).

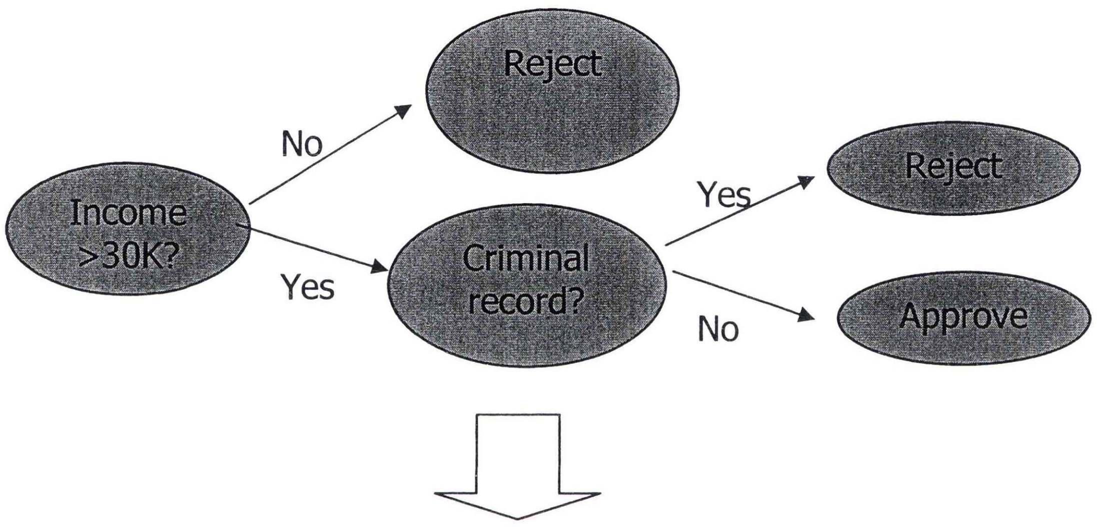
Thus, cost-sensitive learning algorithms should make use of only known values. Of course, the learners may not know exactly how the known values were acquired – were all of them necessary for prediction? In any case, we can assume that the known values may be useful for prediction, but the unknown values are certainly not. Thus, under cost-sensitive learning, there is no need to impute values of any missing data, and the learning algorithms should make use of only known values and that "missing is useful" to minimize the total cost of tests and misclassifications.

5.1.2 Missing or Absent?

In traditional learning algorithm, the unknown data are usually treated as "missing data". However, in the domain of test cost sensitive learning, the situation is different.

In some real application, some fields are usually left as blank because those fields are useless for final decision or too expensive. In this situation, the unknown fields are not used for final decision but they are also different from the traditional "missing values" that are unknown for some unexpected reasons. Those fields are purposely left as unknown, and we call them "absent values".

Examples 5.1: Let's recall the example 2.1 about the credit card application. If the company usually processes the application with the decision in Figure 2.1, then the information of applicant's criminal record is no used for decision in case of his income<30K. Then a historic dataset will be generated as shown in Table 7.1



Ref. Number	Income	Criminal record	Decision
001	15k		Reject
002	26k		Reject
003	40k	Yes	Reject
005	40k	No	Approve
005	40k	No	Approve
	....		

Table 5.1 A Dataset with absent data generated from Figure 2.1.

Previous work of Imputation method is trying to fix all the unknown values and seek the "original" values according to specific criteria, such as statistics measure.

However, in domain of cost-sensitive learning, such as medical diagnostic, doctor usually needs part of the entire patient's medical information and makes a diagnosis. It means other information is left as "absent". Previous work [ZQLS05] shows the absent values are also useful for learning of the doctor's diagnostic knowledge. Here arises a question: do we need to assign a value to the missing field that is actually an "absent" value? On the other hand, all minimal overall cost is the most important issue, i.e. a value that can minimize total cost is preferred than the "best" value upon common sense.

5.2 Review of Previous Work

This chapter deals with missing values in cost-sensitive learning. Turney [Tur00] presents an excellent survey on different types of costs in cost-sensitive learning, among which misclassification costs and test costs are singled out as most important. Much work has been done in recent years on non-uniform misclassification costs (alone), such as [Dom99, Elk01 and Kai98]. Some previous work, such as [Nun91, Tan93], considers the test cost alone without incorporating misclassification cost, which is obviously an oversight. A few researchers [CDYL04, GGD02, Tur95, ZD02] consider both misclassification and test costs, but their methods are less computationally efficient as our approach is based on decision trees. Ling et al. [LYWZ04] propose a decision-tree learning algorithm that uses minimum total cost of tests and misclassifications as the attribute split criterion, and it is the basis of the four missing-value strategies to be presented in Section 3.3. Basically, given a set of training examples, the total cost without further splitting and the total cost after splitting on an attribute can be calculated, and the difference of the two is called cost reduction. The attribute with the maximum, positive cost reduction is chosen for growing the tree. All examples with missing values of an attribute stay at the internal node of that attribute. The method produces decision trees with the minimal total cost of tests and misclassifications on the training data [LYWZ04].

In the next section we will discuss several different missing-value strategies, all of which use the maximum cost reduction strategy described above to build cost-sensitive decision trees.

5.3 Dealing with Missing Values in Cost-sensitive Decision Trees

As we discussed in the Introduction, in cost-sensitive learning which attempts to minimize the total cost of tests and misclassifications, missing data can be useful for cost reduction, and imputing missing values should be unnecessary. Thus, cost-sensitive decision tree learning algorithms should utilize only known values. In the following subsections we will describe four such missing-value techniques. These strategies have been proposed previously but their performance in cost-sensitive learning has not been studied. In Section 6.4 we will perform empirical experiments to compare the four strategies on real-world datasets by the total cost.

5.3.1 The Known Value Strategy

The first tree building and test strategy for "missing is useful" is called the Known Value Strategy. It utilizes only the known attribute values in the tree building for each test example. For each test example, a new (and probably different) decision tree is built from the training examples with only those attributes whose values are known in the test example. That is, the new decision tree only uses attributes with known values in the test example, and thus, when the tree classifies the test example, it will never encounter any missing values.

The Known Value Strategy was proposed in [LYWZ04] but its ability of handling unknown values was not studied. Clearly, the strategy utilizes all known attributes and avoids any missing data directly. It is a lazy tree method [FYK96] where a tree is built during test process. The main drawback of the Known Value Strategy is its relatively high computation cost as different trees may be built for different test examples. This is usually not a problem as the tree building process is very efficient. In addition, we can save frequent trees and use them directly in testing for test examples with the same subsets of known attributes, because decision trees for the same subsets of known attributes are the same. We can use space to trade-off the speed if necessary.

5.3.2 The Null Strategy

As values are missing for a certain reason – unnecessary and too expensive to test – it might be a good idea to assign a special value, often called "null" in databases [DD89], to missing data. The null value is then treated just as a regular known value in the tree building and test processes. This strategy has also been proposed in machine learning [APH93], but its ability in cost-sensitive learning has not been studied.

One potential problem with the Null Strategy is that it does not deliberately utilize the known values, as missing values are treated just as a known value. Another potential drawback is that there might be more than one situation where values are missing. Replacing all missing values by one value (null) may not be adequate. In addition, subtrees can be built under the "null" branch, suggesting oddly that the unknown is more discriminating than known values. The advantage of this strategy is its simplicity and high efficiency compared to the Known Value Strategy, as only one decision tree is built for all test examples.

5.3.3 The Internal Node Strategy

This strategy, as proposed in [LYWZ04] and reviewed in Section 5.2, keeps examples with missing values in internal nodes, and does not build branches for them during tree building. When classifying a test example, if the tree encounters an attribute whose value is unknown, then the class probability of training examples falling at the internal node is used to classify it. As unknown values are dealt with using internal nodes, we call this strategy the Internal Node Strategy.

As there might be several different situations where values are missing, leaving the classification to the internal nodes may be a natural choice. This strategy is also quite efficient as only one tree is built for all test examples.

5.3.4 The C4.5 Strategy

C4.5 [Qui89, Qui93] does not impute missing values explicitly, and it is shown to be quite effective [BM03]. Here C4.5's missing-value strategy is applied directly in cost-sensitive trees. During training, an attribute is chosen by the maximum cost reduction discounted by the probability of missing values of that attribute. During testing, a test example with missing value is split into branches according to the portions of training examples falling into those branches, and goes down to leaves simultaneously. The class of the test example is the weighted classification of all leaves.

5.4 Evaluating and patching up missing values from absent values with hybrid lazy tree

As shown in the last section, absent data is different from missing values. They either are trivial or too expensive to obtain in domain of cost-sensitive learning. Base on this assumption, we could evaluate the unknown values according to their contribution on cost reduction. Those unknown values with trivial contribution obviously are not useful for final decision. We only need to patch or handle the rest ones that are important for final decision. The first problem is how to identify the "absent" values from the missing values. At the same time, some values are known already and they are important for the identifying. The second problem is how to patch up the missing values with the known information.

Traditional imputation methods use value distribution, relation crossing attributes to predict the known values. However, it does not work well in the domain of cost-sensitive learning model, especially in case of matrix with very skew setting. An important reason is they didn't consider the information in cost matrix. In domain of cost-sensitive learning, cost matrix is a very important information source and the performance is measured by overall cost. This leads us to propose a cost-based estimation technique to decide whether the unknown values are missing or absent, and select the proper value which can minimize the misclassification cost for the missing fields.

Here, the *hybrid lazy test cost sensitive decision tree* proposed in chapter 4 is chosen as a promised candidate. The proposed cost sensitive estimation method is different from traditional 0/1 loss imputation method by involving cost matrix. This method is composed of two steps: identifying and patching. For a labeled example in the training set, assume it class label is *L*, and some of its values are known, we need to identify and patch up the missing values from all the known values.

5.4.1 Identifying missing data from absent data

Firstly, we build a lazy test cost-sensitive decision tree *lazyDT* for this example with the revised cost information table (see Section 4.2.2 for detailed descriptions). As mentioned before, absent values are trivial for the final decision. We could make a decision already with the lazy tree *lazyDT* already. So all the attributes not appeared in *lazyDT* will be regarded as "absent"

5.4.2 Patching up missing data

For those attributes with missing values, their corresponding nodes in the *lazyDT* are with two or more branches. Each possible value is corresponding to one branch and we need to choose one value to patch up the missing value. Our patching method is similar with tradition methods that use class label and known values to "guess" the missing values, however the cost matrix is involved to evaluate all possible values. This method chooses a value with helping in reducing overall cost rather than the "best" value upon common sense.

For an internal node with unknown attribute, each branch (or child node) is with a *potential class label L'* (definition 3.3 in Section 3.3.3), if the *L'* is same as *L*, the value of branch is treated as a candidate value, otherwise the value is ignored. For all the candidate values, we sort them by their *potential label misclassification cost* (definition 3.3 in Section 3.3.3). The candidate value with minimal *potential label misclassification cost* is select to patch the missing field.

Examples 5.2: Assume in a binary classification problem, the misclassification cost FP= 200, FN =600. An example instance with unknown values is shown in Table 5.2. We can see that *A1, A2, A4,* and *A5* are the only known attributes, and a hybrid lazy test sensitive decision tree is built as in Figure5.1.

Attribute	Al	A2	A3	A4	A5	A6	Class
Value	6	2	? (1)	2	2	? (3)	P
Test cost	0	0	50	0	0	20	

Table 5.2 An example with unknown values and new test costs

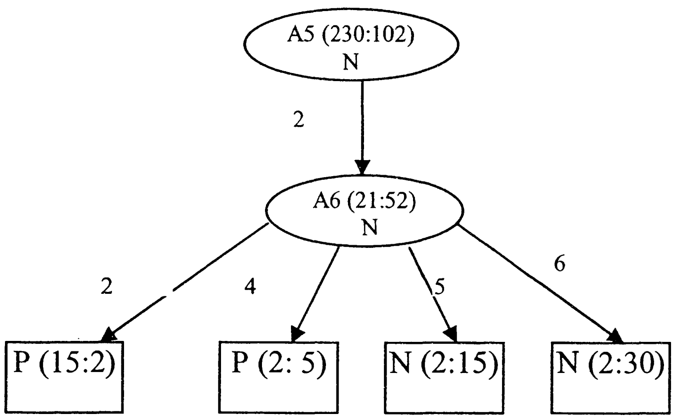


Figure 5.1. A decision tree built extended from figure 7.1.

For the two unknown attributes *A3 and A6*, only *A6* is selected into the tree. That means *A3* is trivial for decision making and should be marked as “absent”. We only need to patch up the missing value of attribute *A6.*

For the node of attribute *A6,* only the branches for values 2 and 4 are marked as class label *P*, so we only choose a value from the two candidates. Now we calculate the *potential label misclassification cost* for the two branches. Here, marking branch of value 2 as *P* incurs misclassification cost 2*200 = 400; marking branch of value 2 as *P* incurs misclassification cost 5*200 = 1000. Obviously, the first branch is with lesser misclassification cost, so we fill the missing field of A6 as value "2" in Table 5.2.

5.5 Experiments

In this section, we conducted experiments with same datasets as in section 3.5. We compare the four missing-value strategies discussed in Section 5.3. Then we study the performance of proposed strategies about dealing with absent values.

5.5.1 Comparing the Four Missing-value Strategies

We randomly assign random numbers between 0 and 100 to each attribute as test costs. We also assign 200 for false positive, and 600 for false negative misclassification costs. The cost of true positives and true negatives is set to 0. These assumptions are reasonable as attributes do have some costs in real world, and we compare the four missing-value strategies based on the same test and misclassification costs.

To simulate missing values in datasets, we randomly select certain percentages (20%, 40%, 60%, and 80%) of attribute values in the whole dataset to be missing, and those missing values are distributed into each attribute proportional to its cost, as more expensive attributes usually have more missing values. Each dataset is then split into training and test sets using 10-fold cross validation (thus test sets also have the same percentages of missing values). For each split, a decision tree is built from the training dataset, and is applied to the test examples, using the Null Strategy, the Internal Node Strategy, and the C4.5 Strategy. For the Known Value Strategy, a lazy tree is built for each test example. The full experimental results for the four missing-value strategies are listed in Table 5.3:

	20%	40%	60%	80%
The Ecoli Dataset				
The Known Value Strategy	135.1	144.5	134.2	125.8
The Null Strategy	28.3	33.9	41.4	42.5
The Internal Node Strategy	33.2	48.6	53.5	62.3
The C4.5 Strategy	35.0	42.5	58.9	72.6
The Breast Dataset				
The Known Value Strategy	67.6	91.9	111.3	116.2
The Null Strategy	53.3	61.4	69.8	74.2
The Internal Node Strategy	51.0	59.6	63.5	77.3
The C4.5 Strategy	52.2	57.8	72.6	71.4
The Heart Dataset				
The Known Value Strategy	146.6	126.0	98.2	121.9
The Null Strategy	90.3	88.6	103.7	98.8
The Internal Node Strategy	86.6	85.3	83.2	88.2
The C4.5 Strategy	88.2	87.6	83.2	88.9
The Thyroid Dataset				
The Known Value Strategy	169.4	153.7	138.9	108.5
The Null Strategy	66.6	72.7	76.1	73.3
The Internal Node Strategy	64.4	70.7	71.8	71.7
The C4.5 Strategy	64.4	72.3	90.5	72.4
The Australia Dataset				
The Known Value Strategy	174.2	143.0	106.3	107.4
The Null Strategy	115.3	99.2	121.0	113.1
The Internal Node Strategy	97.1	90.7	94.4	96.8
The C4.5 Strategy	98.1	94.0	109.4	96.2

Table 5.3 Experiment results for four missing-value strategies

The performance of the four missing-value strategies is measured by the average total cost of tests and misclassifications of test examples in the 10-fold cross-validation. Here the test cost is the total cost of the tests (attributes) in actually classifying test examples. That is, it is the "effective" test cost, not the sum of test costs of known attributes in test examples. As we discussed in Section 5.1, some tests may be unnecessary for prediction, as doctors may subscribe more tests than needed for diagnosis. Therefore we use the "effective" test cost to better measure each strategy's actual performance. The misclassification cost is calculated as usual: if the prediction is correct, the misclassification cost is 0; otherwise, it is either the false positive cost or false negative cost, depending on the true class of the test examples. Table 5.3 lists the average total cost with different missing-value strategies under different percentages of missing values in the datasets. Figures 5.3 to 5.7 have illustrated the results in Table 5.3 visually.

We can draw the following interesting conclusions from the results. First of all, the Known Value Strategy (KV) is almost always the worst. This is because deleting attributes with missing values in the test example makes useful information lost in the datasets. Thus, this strategy should be avoided in the future. Second, in only one dataset (Ecoli) the Null Strategy is slightly better than others; in other datasets, it is either similar (in Breast and Thyroid) or worse (in Heart and Australia). This shows that the Null Strategy, although very simple, is often not suitable. Third, the Internal Node Strategy is often comparable with the C4.5 Strategy (in Ecoli, Breast, and Heart) and is better than C4.5 in Thyroid and Australia. This indicates that, overall, the Internal Node Strategy is better than C4.5. Thus, we can conclude from our experiments that the Internal Node Strategy is the best, closely followed by the C4.5 Strategy, and followed by the Null Strategy. The Known Value Strategy is the worst.

It might be slightly counterintuitive why the C4.5 Strategy, which obtains weighted classifications from leaves, is not better than the Internet Node Strategy that uses the internal node directly. This is because when it weighs leave's classifications, there is a loss of information. If it weighs the leaves' probabilities, it can be shown easily that the result is equivalent to the class probability in the internal node in the Internal Node Strategy. Thus, the Internal Node Strategy is better than the C4.5 Strategy.

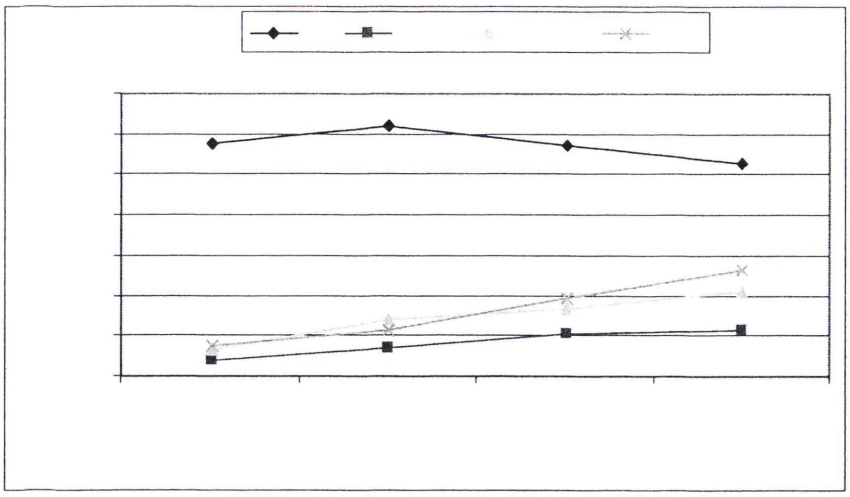


Figure 5.2 Total average costs for Ecoli. In this and the following figures, "KV" stands for the Known Value Strategy, "NULL" for the Null Strategy, "Internal" for the Internal Node Strategy, and "C4.5" for the C4.5 Strategy.

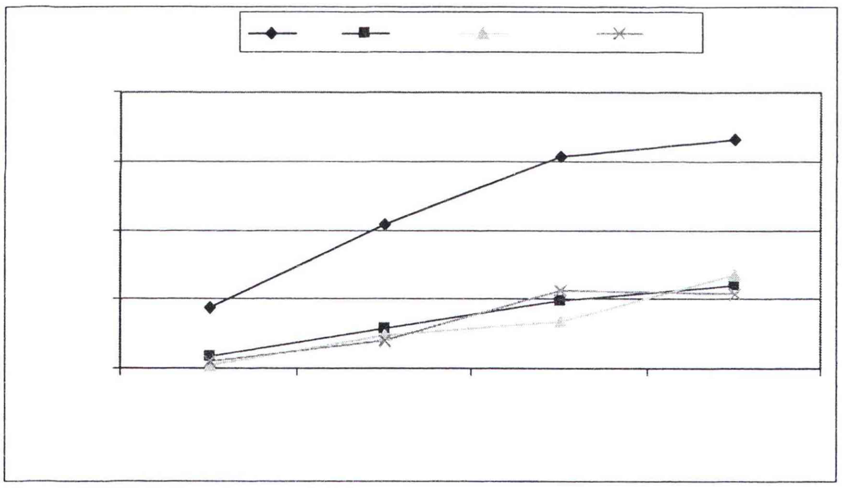


Figure 5.3 Total average costs for Breast

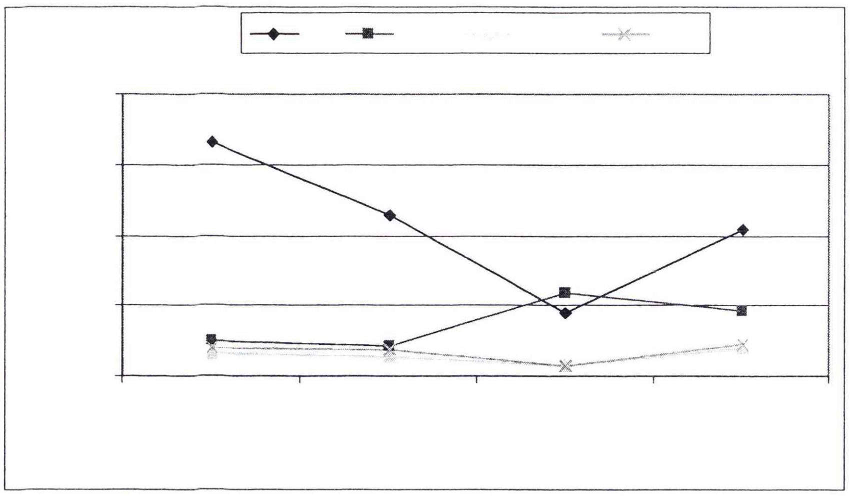


Figure 5.4 Total average costs for Heart

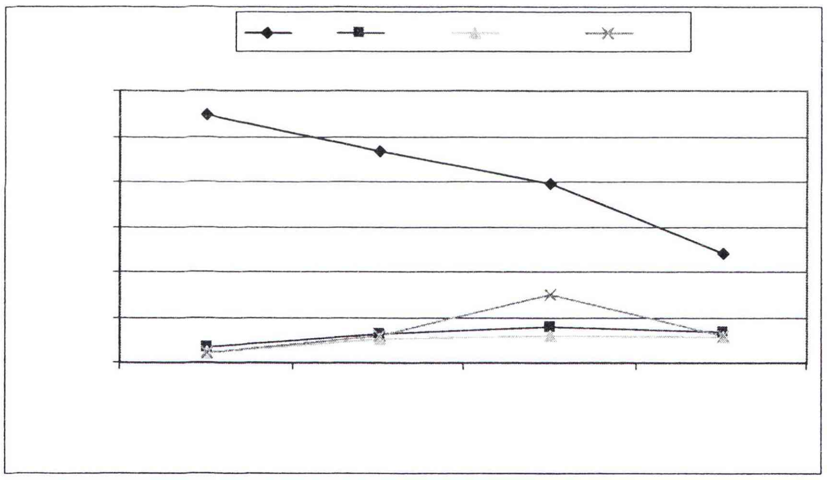


Figure 5.5 Total average costs for Thyroid

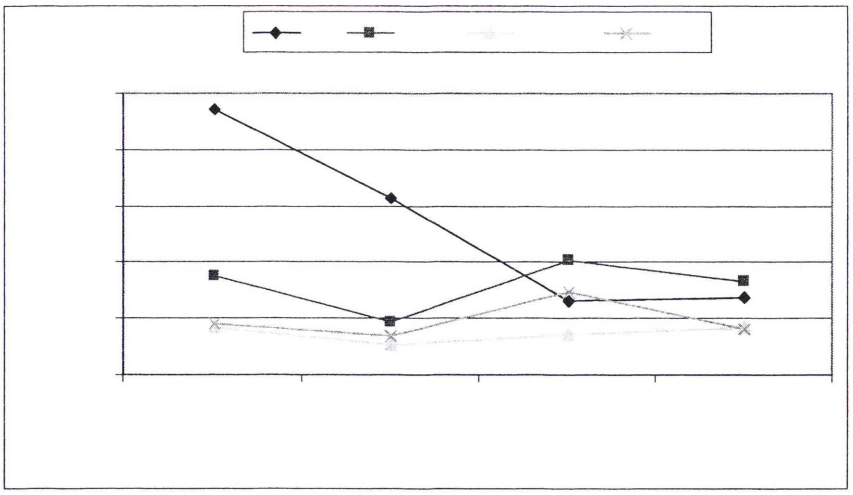


Figure 5.6 Total average costs for Australia

5.5.2 Experiments for identifying absent data

We compare the average cost (Total Cost/ Example number in test phase) of our evaluating and patching strategy with C4.5 strategy and internal node strategy. The C4.5 strategy treats all known values as missing values and uses the probability of values to "guess" the real values. The internal node strategy treats all the known values as "absent values" and directly leaning the original training data.

First we randomly choose 5% values from the dataset D, new dataset noted by D'. Using our patching method, D' is evaluated and patched, patched dataset noted by D''. And we conducted the tree building algorithm in [LYWZ04] on both dataset D' and D''. Performance of the two methods is also compared to C4.5 conducting on D'. The results of total cost on all four datasets of Section 3.5 are shown as in Figure 5.8. From Figure 5.8, we can see that patching method outperforms the other two in total cost. It means building decision on patched dataset has got a better overall performance than building on dataset with imperfection.

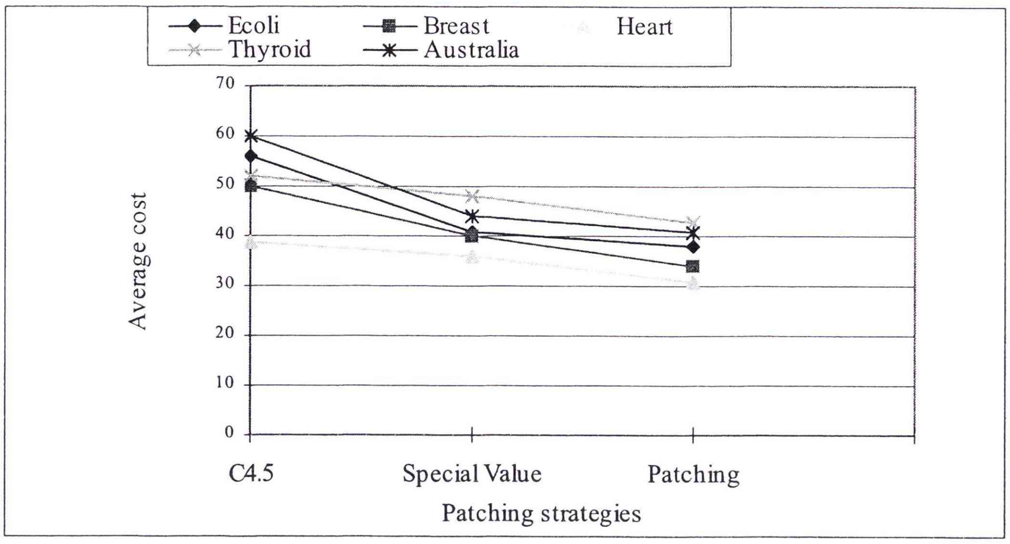


Figure 5.7 Comparing of average cost of three handling strategies on all datasets

To study the influence of cost matrix on choosing values, we conducted patching on different cost matrix (FP/FN from 800:800 to 200:800). The result is shown in Figure 5.9. From Figure 7.6, we can see that cost matrix will significantly influence the patching performance. And our method work better on skew cost. We can conclude that cost matrix is very important for patching missing values in cost sensitive learning.

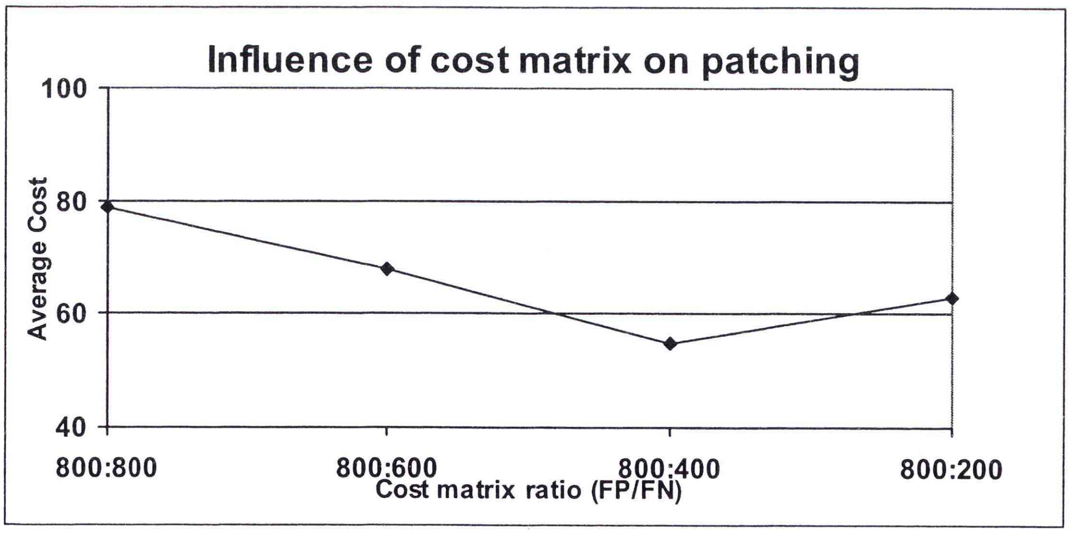


Figure 5.8 Influence of cost matrix on our patching model

5.6 Conclusions and Future Work

Missing values are traditionally regarded as a tough problem, and must be imputed before learning is applied. In this chapter we break away from this tradition, and argue that in cost-sensitive learning that also considers test costs, it is actually desirable to have some unknown values to reduce the total cost of tests and misclassifications. Those unknown values are noted as “absent values”. Thus, cost-sensitive decision tree learning algorithms would only need the known values, and take advantage of “absent is useful” for cost reduction. We compare four such strategies, and conclude that the Internet Node Strategy, originally proposed in [LYWZ04], is the best.

We present a hybrid lazy tree based strategy to evaluate and patch the unknown values in cost-sensitive learning. To minimize the misclassification cost, value with least risk of high cost is selected to fill the missing fields. We proposed a new cost-based estimation criterion for value selection, and only the “important” values are imputed and others are marked as “absent”. Our experiments show that our cost-based evaluating and patching strategy outperforms the existing strategies in learning the data with known values.

In our Future work, we plan to do further research on absent data and missing data in test cost sensitive learning. We will study the essential properties of absent data and missing data. We plan to apply strategies in section 5.3 to datasets with real costs and missing values, especially in the medical diagnosis domains.

We also plan to design better strategies for identifying and patching the missing values on test sensitive decision tree and other cost-sensitive learners, such as Naive

Bayes classifiers. Also pruning can be introduced in our tree-building algorithm to avoid over-fitting of the data.

Chapter 6 Conclusions and Future Research

Recently, researchers have begun to consider both attribute test cost and misclassification cost in cost sensitive learning. Previous work assumes the test cost and the misclassification cost must be defined on the same cost scale, such as the dollar cost incurred in a medical diagnosis. And they aim to minimize the expected total cost of the misclassification cost and the attribute test cost. However, this is the only way to combine the misclassification cost and the attribute test cost together? Detailed studies are needed to ensure the ways of combination make sense and be "correct", dimensionally as well as semantically.

This dissertation argues for involving both test cost and misclassification cost, or even more costs together, and performs further studies on fundamental properties of attribute test costs and proposes a new way to combine the two costs together. This chapter concludes the dissertation by outlining proposed key techniques and looking towards the future research.

6.1 Conclusions

i) A new CSL problem is presented as meeting with difficulty in defining multiple costs on a certain cost scale. To tackle this issue, a *target-resource budget learning framework* with multiple costs is proposed. With this framework, we present a test cost sensitive decision tree model with two kinds of cost scales. The task is to minimize one cost scale, called target cost, and keep the other one within specified budgets. To the best of our knowledge, this is the first one to study the cost sensitive learning with multiple costs scales.

ii) A lazy test sensitive decision tree model with multiple cost scales is proposed. This model is based on the assumption that some attributes of an unlabeled example are known before being classified. With this model, we could utilize the known information and reduce the overall cost. We also modify and apply this model to the batch-test problem: multiple tests are chosen and done in one shot, rather than in a sequential manner in the test-sensitive tree. It is significant in some diagnosis applications that require making a decision as soon as possible.

iii) Some essential properties of attribute test cost are studied. This is the first work to utilize the information hidden in those "absent data" in cost sensitive learning; and the conclusion is very positive, i.e. "absent data" is also useful for decision making. This thesis also studies the difference between the 'absent data' and 'missing data', i.e. 'absent data' is trivial and we could just use other data to make a decision; whereas 'missing data' may be important for decision making and we should patch them up before performing our learning task. A lazy decision tree based algorithm is proposed to identify the absent data from missing data, and a novel strategy is proposed to help patch the "real" missing values.

These techniques are very different from traditional test cost sensitive learning. They focus on some fundamental issues on cost and their combinations that are not noticed by other researches. However, those issues and properties are very important for cost sensitive theory and technology. Extensive experiments are conducted for evaluating the proposed approaches, and have demonstrated that the work in this dissertation is efficient. There are four positive features of the techniques proposed in this dissertation.

(1) **Effectiveness.** As we see in each chapter, all the proposed techniques aim to reduce the overall cost of cost sensitive and draw very positive conclusion. Our methods usually outperform existing ones on most of cases with cost analysis sensitive benchmark.

(2) **Generalization**. Most of our techniques start from existing methods and aim to solve existing problems with those methods. Such as the target-resource budget framework in Chapter 3, existing test cost sensitive learning is only a special case of proposed framework.

(3) **Novelty.** All the work is original and novel. The target-resource budget framework in Chapter 3 is the first work on cost sensitive learning with multiple cost scales. And we also are the first ones who propose a definition and utilization of "absent data"

(4) **Verification.** All the proposed techniques have been evaluated by empirical experiment study.

6.2 Future Research

Costs and their combination are the core issues in cost sensitive learning theory and technology. This dissertation conducts some fundamental studies on the properties of costs involved and new ways to combine them together. However, it is still a challenging task in this area.

6.2.1 Combination of Multiple costs

In Chapter 3, the proposed target-resource budget framework could handle multiple costs. However, with current proceeding in cost sensitive learning, we only conduct extensive studies on the case of combining both of attribute test cost and misclassification cost. Combination of three or more kinds of costs into a cost sensitive learning model is a challenging task, especially in the case when those costs cannot be defined in same cost scale.

In the future, we plan to extend our studies on more complex situations. In the next step, we plan to conduct further studies on combination of both of teaching cost and misclassification cost. For example, we could consider how to minimize the total target cost with partial cost-resource exchanging. In some situations, such as medical diagnosis, this scenario is more practical since many hospitals provide VIP services. We also want to extend our test strategy to near Optimal Batch Test, and pruning can be introduced in our tree-building algorithm to avoid over-fitting of the data.

6.2.2 Properties of other costs

Most of the existing work on costs and their properties are concentrated on attribute test cost and misclassification cost. For instance, the "absent data" arises because we only need to perform part of tests and make a satisfactory decision. In the future, we will conduct further studies on the properties of test cost and misclassification cost. We could think about the properties of other costs, such as teaching cost.

We plan to verify the conclusions on datasets with real costs and missing values. We also plan to test the patching method on other cost-sensitive learners, such as Naive

Bayes. Also pruning can be introduced in our tree-building algorithm to avoid over-fitting of the data.

Bibliography

[AIS93] R.Agrawal, T.Imielinski, and A. Swami, Database mining: a performance perspective. *IEEE Transactions Knowledge and Data Engneering*, 5(6) 1993: 914-925.

[APH93] K. M. Ali and M. J. Pazzani, Hydra: A noise-tolerant relational concept learning algorithm. In: *Proceedings of the 13th International Joint Conference on Artificial Intelligence* (IJCAI93), pp. 1064-1071. Morgan Kaufmann, 1993.

[All02] P.D. Allison, *Missing data.* Thousand Oaks, CA: Sage 2002.

[AZ05] A. Arnt, S. Zilberstein, Learning Policies for Sequential Time and Cost Sensitive Classification. In: *First International Workshop on Utility-Based Data Mining in KDD2005*, pp. 39-45, August 21, 2005 Chicago, Illinois, USA.

[BM03] G. Batista and M. C. Monard, An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence*, Vol. 17, pp. 519-533, 2003.

[BM98] C. L. Blake and C. J. Merz, *UCI Repository of machine learning databases.* (See [http://www.ics.uci.edu/~mlearn/MLRepository.html]). Irvine, CA: University of California, Department of Information and Computer Science, 1998.

[BKKB98] J. P. Bradford, C. Kunz, R. Kohavi, C. Brunk, and C. E. Brodley, Pruning decision trees with misclassification costs. In: *European Conference on Machine*

Learning, pp. 131-136, Longer version from ECE TR 98- 3, Purdue University, 1998 or http://robotics.stanford.edu/users/ronnyk/ronnyk-bib.html.

[BFOS84] L. Breiman, J. H. Friedman, R. H. Olshen, and C. J Stone, *Classification and regression trees*. Wadsworth, Belmont, California, 1984.

[CDYL04] X. Chai, L. Deng, Q. Yang, and C. X. Ling, Test-cost sensitive naïve Bayesian classification. In: *Proceedings of the Fourth IEEE International Conference on Data Mining* (ICDM04), Brighton, UK: IEEE Computer Society Press, 2004.

[CS95] P. Cheeseman, and J. Stutz, Bayesian classification (AutoClass): Theory and results. In: *Advances in Knowledge Discovery and Data Mining*, pp. 153-180, AAAI Press, Menlo Park, CA, 1995.

[CN89] P. Clark and T. Niblett, The CN2 induction algorithm. *Machine Learning*, Vol. 3, pp. 261–283, 1989.

[CT91] T. M. Cover, and J.A. Thomas, *Elements of Information Theory*. John Wiley &Sons, Inc. 1989.

[DD89] C. J. Date and H. Darwen, The default values approach to missing information. *Relational Database Writings 1989-1991*, pp. 343-354, 1989.

[Dom99] P. Domingos, MetaCost: A general method for making classifiers cost-sensitive. In: *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining* (KDD99), pp. 155-164. San Diego, CA: ACM Press, 1999.

[DP00] P. Domingos and F. Provost, Well-trained PETs: *Improving probability estimation trees*. In: CDER Working Paper #00-04-IS, Stern School of Business, New York University, New York, 2000.

[Elk01] C. Elkan, The foundations of cost-sensitive learning. In: *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence* (IJCAI01), pp. 973-978, 2001.

[FP96] T. Fawcett, and F.J. Provost, Combining data mining and machine learning for effective user profiling. In: *Proceedings of the Second International Conference on* Knowledge Discovery and Data Mining (KDD96), pp. 8-13. 1996

[FP97] T. Fawcett, and F.J. Provost, Adaptive fraud detection. Data Mining and Knowledge Discovery, 1 (3), 1997

[FI93] U. M. Fayyad and K.B. Irani, Multi-interval discretization of continuous-valued attributes for classification learning. In: *Proceedings of the 13th International Joint Conference on Artificial Intelligence* (IJCAI93), pp. 1022-1027. Morgan Kaufmann, 1993.

[FS81] J. H. Friedman and W. Stuetzle, Projection pursuit regression. *Journal of the American Statistics Association*, 76:817-823, 1981

[FYK96] J. Friedman, Y. Yun and R. Kohavi, Lazy decision trees. In: *Proceedings of the 13th National Conference Artificial Intelligence* (AAAI96), pp. 717-724, 1996.

[Gam00] J. Gama, A cost-sensitive iterative Bayes. In: *Workshop on Cost-Sensitive Learning at ICML2000*, pages 7{13, Stanford University, California, 2000

[GP89] D. Gordon and D. Perlis, Explicitly biased generalization. *Computational Intelligence*, 5(2):67-81, 1989.

[GGR02] R. Greiner, A. J. Grove, and D. Roth, Learning cost-sensitive active classifiers. *Artificial Intelligence*, 139(2): 137-174, 2002.

[HK00] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. The Morgan Kaufmann Series in Data Management Systems, Jim Gray, Series Editor, Morgan Kaufmann Publishers, August 2000. 550 pages, ISBN 1-55860-489-8.

[HHB74] J. Hermans, J.D.F. Habbema, and, V. D. Burght, Cases of doubt in allocation problems, K populations. *Bulletin of the International Statistics Institute*, 45, 523-529. 1974.

[HZ02] X. Huang and Q. Zhu, A Pseudo Nearest-Neighbor Substitution Approach for Missing Data Recovery on Gaussian Random Data Sets. *Pattern Recognition Letters*, Vol. 23, No. 13, pp. 1613-1622, 2002.

[Kai98] M. T. Kai, Inducing Cost-sensitive trees via instance weighting. In: *Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery*, pp. 23-26, Springer-Verlag, 1998.

[KNT94] U. Knoll, G. Nakhaeizadeh, and B. Tausend, Cost-sensitive pruning of decision trees. In: *Proceedings of the Eighth European Conference of Machine Learning*, pages 383-386, Berlin, Germany, 1994, Springer-Verlag.

[KK98] M. Kukar and I. Kononenko, Cost-sensitive learning with neural networks. In: *Proceedings of the Thirteenth European Conference Artificial Intelligence*, pages 445-449, Chichester, 1998. John Wiley & Sons, New York.

[Lan96] P. Langley, *Elements of Machine Learning.* Morgan Kaufmann, San Mateo, California, 1996.

[LC87] P. Langley, J.G. Carbonell, Machine learning. *Encyclopedia of artificial intelligence*, New York: John Wiley & Sons.

[Lar05] D. T. Larose, *Discovering Knowledge in Data* – An Introduction to Data Mining. John Wiley & Son, Inc., Hoboken, New Jersey, 2005

[LHS99] K. Lakshminarayan, S. A. Harp and T. Samad, Imputation of missing data in industrial databases. *Applied Intelligence*, Vol. 11, pp. 259-275, 1999.

[LSY06] C. X. Ling, S. Sheng, Q. Yang, Test Strategies for Cost-Sensitive Decision Trees. *IEEE Transactions on Knowledge and Data Engineering* (TKDE), to appear.

[LYWZ04] C. Ling, Q. Yang, J. Wang and S. Zhang, Decision Trees with Minimal Costs. In: *Proceedings of 21st International Conference on Machine Learning* (ICML04), Banff, Alberta, Canada, July 4-8, 2004.

[LR87] R. J. A. Little and D. B. Rubin, *Statistical analysis with missing data.* New York: John Wiley, 1987.

[LNHFH03] M. L. Littman, T. Nguyen, H. Hirsh, E. M. Fenson, and R. Howard, Cost sensitive fault remediation for autonomic computing. In: *Workshop on AI and Autonomic Computing: Developing a Research Agenda for Self-Managing Computer Systems*, 2003.

[LW97] W. Z Liu, A. P White, Techniques for dealing with missing values in classification. In: *Second International Symposium on Intelligent Data Analysis*, London, 1997

[Mar01] D. D. Margineantu, Methods for Cost-sensitive Learning. PhD thesis, Department of Computer Science, Oregon State University, Corvallis, 2001.

[MD02] D. D. Margineantu and T. G. Dietterich, Improved class probability estimates from decision tree models. In: Nonlinear Estimation *and Classification*, Lecture Notes in Statistics, volume 171, pages 169-184, New York, 2002. Springer-Verlag

[Mit97] T. M Mitchell, *Machine Learning*. McGraw Hillsds, 1997.

[Nor89] S. W. Norton, Generating better decision trees. In: Proceedings of the Eleventh In: *International Joint Conference on Artificial Intelligence*, pages 800-805, San Francisco, 1989. Morgan Kaufmann.

[Nun91] M. Nunez, The use of background knowledge in decision tree induction. *Machine Learning*, Vol. 6, pp. 231-250, 1991.

[PH90] G. Pagallo, & D. Haussler, Boolean feature discovery in empirical learning. *Machine Learning*, Vol. 5, 71-99, 1990.

[PA90] K. R. Pattipati and M. G. Alexandridis, Application of heuristic search and information theory to sequential fault diagnosis. *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 20(4): 872-887, 1990.

[PMMA94] M. Pazzani, C. Merz, P. Murphy, K. Ali, T. Hume, and C. Brunk, Reducing misclassification costs. In: *Proceedings of the Eleventh International Conference of Machine Learning*, pp. 217-225, New Brunswick, New Jersey, 1994. Morgan Kaufmann.

[PF97] F. J. Provost and T. Fawcett, Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In: *Knowledge Discovery and Data Mining*, pp. 43-48, 1997.

[PF01] F. J. Provost and T. Fawcett, Robust classification for imprecise environments. *Machine Learning*, vol. 42(3), pp. 203-231, 2001.

[QLZ04] Z. Qin, L. Liu and S. Zhang, Mining Term Association Rules for Heuristic Query Construction. In: *Proceedings of 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining* (PAKDD04) Sydney, Australia, May 26-28, pp.145-154. 2004.

[QZZ04] Z. Qin, C. Zhang and S. Zhang, Cost-sensitive Decision Trees with Multiple Cost Scales. In: *Proceedings of the 17th Australian Joint Conference on Artificial Intelligence* (AI 2004), Cairns, Queensland, Australia, 6-10 December, 2004.

[QZZ05] Z. Qin, C. Zhang and S. Zhang, Dynamic Test-sensitive Decision Trees with Multiple Cost Scales. In: *Proceedings of International Conference on Fuzzy Systems and Knowledge Discovery* (FSKD2005), pp. 402-405, Changsha, China, August 2005

[QZZ06] Z. Qin, C. Zhang and S. Zhang, Missing or absent? A Question in Cost-sensitive Decision Tree. In: *Proceedings of the Fourth International Conference on Active Media Technology* (AMT2006), Jun. 2006:

[Qui89] J. R. Quinlan, Unknown attribute values in induction. In: *Proceeding Of the Sixth International Workshop on Machin Learning*, pp. 164-168, Morgan Kaufmann, Los Altos, USA, 1989.

[Qui93] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann. San Mateo, California, 1993.

[RC99] A. Ragel and B. Cremilleux, MVC, A preprocessing Method to deal with missing values. *Knowledge-based Systems*, pp. 285-291, 1999.

[SL06] V. S. Sheng and C. X. Ling, Feature Value Acquisition in Testing: A Sequential Batch Test Algorithm. In: *Proceedings of 2006 International Conference on Machine Learning* (ICML06), 2006.

[SLNZ06] V. S. Sheng, C. X. Ling, A. Ni, and S. Zhang, Cost-Sensitive Test Strategies. In: *Proceedings of the Twenty-first National Conference on Artificial Intelligence* (AAAI06), 2006.

[SLY05] S. Sheng, C. X. Ling, Q. Yang, Simple Test Strategies for Cost-Sensitive Decision Trees. In: *Proceedings of the 16th European Conference on Machine Learning* (ECML), 2005.

[STV97] M.W. Someren, C. Torres, F. Verdenius, A systematic description of greedy optimization algorithms for cost sensitive generalization. In: *Proceedings of Intelligent Data Analysis* 1997 (IDA97), pp. 247-258, New York, 1997, Springer Verlag.

[Tan93] M. Tan, Cost-sensitive learning of classification knowledge and its applications in robotics. *Machine Learning*, Vol. 13, pp. 7-33, 1993.

[TS90] M. Tan and J. C. Schlimmer. Two case studies in cost-sensitive concept acquisition. In: *Proceedings of the Eighth National Conference on Artificial Intelligence*, pp. 854-860, Menlo Park, California, 1990, AAAI Press.

[Ten04] C. M. Teng, Polishing blemishes: Issues in data correction. *IEEE Intelligent Systems*, pp 34-39, March/April 2004

[Tur00] P. D. Turney, Types of cost in inductive concept learning. In: *Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning*, Stanford University, California, 2000

[Tur95] P. D. Turney, Cost-sensitive classication: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Articial Intelligence Research*, Vol. 2, pp. 369-409, 1995.

[Ver91] F. Verdenius, A method for inductive cost optimization. In: *Proceedings of the Fifth European Working Session on Learning* (EWSL91), pp. 179-191, New York, 1991.

[Web96] G. I. Webb, Cost-sensitive specialization. In: *Proceedings of the 1996 Pacific Rim International Conference on Artificial Intelligence*, pp. 23-34, Springer-Verlag, 1996.

[WGT90] S. M. Weiss, R. S. Galen, and P. V. Tadepalli, Maximizing the predictive value production rules. *Artificial Intelligence*, Vol. 45(1-2):47-71, 1990.

[WF05] I. H. Witten, E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques* (Second Edition). Morgan Kaufmann, 525 pages, 2005

[YLCP06] Q. Yang, C. X. Ling, X. Chai, and R. Pan, Test-Cost Sensitive Classification on Data with Missing Values. *IEEE Transactions on Knowledge and Data Engineering* (TKDE), to appear.

[YYPQ05] Y. Yang, Q. Yang, R. Pan et al. and Z. Qin, Preprocessing Time Series Data for Classification with Application to CRM. In: *Proceedings of the 18th Australian Joint Conference on Artificial Intelligence* (AI2005), pp. 133-142, Sydney, Australia, 2005.

[ZE01] B. Zadrozny and C. Elkan, Learning and making decisions when costs and probabilities are both unknown. In: *Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining*, pp. 204-213, ACM Press, 2001.

[ZQY05] C. Zhang, Z. Qin, X. Yan, Association-Based Segmentation for Chinese-Crossed Query Expansion. *IEEE Intelligent Informatics Bulletin*, Vol. 5(1), pp.18-25, 2005.

[ZZYQ02] C. Zhang, S. Zhang, X. Yan and Z. Qin, Identifying Exceptional Patterns in Multi-databases. In: *Proceedings of the 2002 International Conference on Fuzzy Systems and Knowledge Discovery*, Singapore, pp. 476-480, November 2002.

[ZQLS05] S. Zhang, Z. Qin, C. Ling and S. Sheng, "Missing is Useful": Missing Values in Cost-sensitive Decision Trees. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 17 No. 12, pp. 1689-1693, 2005.

[ZZQ03] S. Zhang, C. Zhang and Z. Qin, Modeling Temporal Semantics of Data. *Asian Journal of Information Technology*, Vol. 2(1) pp. 25-35, 2003.

[ZZW04] S. Zhang, C. Zhang and X. Wu, *Knowledge Discovery in Multiple Databases*. 233 pages, ISBN: 1-85233-703-6, Springer, 2004.

[ZZYQ02] S. Zhang, C. Zhang, X. Yan and Z. Qin, Mind The Trends When You Mine: Incremental Data Mining. In: *Proceedings of the 2002 International Conference on Fuzzy Systems and Knowledge Discovery*, Singapore, pp. 146-150, Nov. 2002.

[Zub2003] V. B. Zubek, *Learning Cost-Sensitive Diagnostic Policies from Data.* Ph.D Dissertation submitted to Oregon State University, 2003.

[ZD02] V. B. Zubek, T. G. Dietterich, Pruning Improves Heuristic Search for Cost-Sensitive Learning. In: *Proceedings of the Nineteenth International Conference on Machine Learning*, pp. 27-34, Sydney, Australia, 2002.