A survey of cost-sensitive decision tree induction algorithms

Lomax, S and Vadera, S

http://dx.doi.org/10.1145/2431211.2431215

A Survey of Cost-Sensitive Decision Tree Induction Algorithms

SUSAN LOMAX, University of Salford
SUNIL VADERA, University of Salford

The past decade has seen a significant interest on the problem of inducing decision trees that take account of costs of misclassification and costs of acquiring the features used for decision making. This survey identifies over 50 algorithms including approaches that are direct adaptations of accuracy based methods, use genetic algorithms, use anytime methods and utilize boosting and bagging. The survey brings together these different studies and novel approaches to cost-sensitive decision tree learning, provides a useful taxonomy, a historical timeline of how the field has developed and should provide a useful reference point for future research in this field.

Categories and Subject Descriptors: **I.2.6 [Artificial Intelligence]:** Learning - induction

General Terms: Algorithms

Additional Key Words and Phrases: Decision Tree Learning, Cost-Sensitive Learning, Data mining

ACM Reference Format:
Lomax, S., and Vadera, S. 2011. A survey of cost-sensitive decision tree induction algorithms. ACM Computing Surveys. Article (), 34 pages.

DOI =

1. INTRODUCTION

Decision trees are a natural way of presenting a decision-making process, because they are simple and easy for anyone to understand [Quinlan 1986]. Learning decision trees from data however is more complex, with most methods based on an algorithm, known as ID3 that was developed by Quinlan [1979, 1983,1986]. ID3 takes a table of examples as input, where each example consists of a collection of attributes, together with an outcome (or class) and induces a decision tree, where each node is a test on an attribute, each branch is the outcome of that test and at the end are leaf nodes indicating the class to which the example, when following that path, belongs. ID3, and a number of its immediate descendents, such as C4.5 [Quinlan 1993], OC1 [Murthy et al. 1994] and CART [Breiman et al. 1984] focused on inducing decision trees that maximized accuracy.

However, several authors have recognized that in practice there are costs involved (e.g. Breimen et al. [1984]; Turney [1995]; Elkan [2001]). For example, it costs time and money for blood tests to be carried out [Quinlan et al. 1987]. In addition, when examples are misclassified, they may incur varying costs of misclassification depending on whether they are false negatives (classifying a positive example as negative) or false positives (classifying a negative example as positive). This has led to many studies that develop algorithms that aim to induce cost-sensitive decision trees. These studies are presented in many different sources and, to the best of our knowledge; there is no comprehensive synthesis of cost-sensitive induction algorithms. Hence, this survey aims to provide an overview of existing algorithms and their characteristics that should be a useful source for any researcher or practitioner seeking to study, develop or apply cost-sensitive decision tree learning.

Section 2 of the paper begins with a brief introduction to decision tree induction to set the context for readers not already familiar with this field. The survey identified over fifty algorithms, some of which are well known and cited, but also some that are less well known. Section 3 begins by presenting a taxonomy of cost-sensitive decision tree algorithms that is based on the algorithms identified. Sections 4 and 5 present a survey of the algorithms based on the taxonomy, and Section 6 concludes the paper.

2. BACKGROUND TO DECISION TREE INDUCTION

Given a set of examples, early decision tree algorithms, such as ID3 and CART, utilize a greedy top-down procedure. An attribute is first selected as the root node using a statistical measure [Quinlan 1979, 1983; Breiman et al. 1984]. The examples are then filtered into subsets according to values of the selected attribute. The same process is then applied recursively to each of the subsets until a stopping condition, such as a certain proportion of examples being of the same class. The leaf nodes are then assigned the majority class as the outcome. Researchers have experimented with different selection measures, such as the GINI index [Breiman et al. 1984], using chi-squared [Hart 1985] and which have been evaluated empirically [Mingers 1989]. The selection measure utilized in ID3 is based on Information Theory which provides a measure of disorder, often referred to as the entropy, and which is used to define the expected entropy, *E* for an attribute A [Shannon 1948; Quinlan 1979; Winston 1993]:

$$E(A) = \sum_{a \in A} P(a).\sum_{c \in C} -\mathrm{P}(a|c)\log 2(\mathrm{P}(a|c))) \qquad (1)$$

where $a \in$ A are the values of attribute A, and the $c \in$ C are the class values.

This formula measures the extent to which the data is homogeneous. For example, if all the data were to belong to the same class, the entropy would be '0'. Likewise if all the examples belonged to different classes, the entropy would be '1'. ID3 uses an extension of the entropy by calculating the gain in information *(I)* achieved by each of the attributes if these were chosen for the split and choosing the attribute which maximizes this gain:

ID3: $I_A = E(D) - E(A)$

where $E(D) = \sum_{c \in C} -\frac{\mathrm{N_c}}{\mathrm{N}} log_2 \frac{\mathrm{N_c}}{\mathrm{N}}$, calculated on the current training set before splitting.

Although, Quinlan adopted this measure for ID3, he noticed that the measure is biased towards attributes that have more values, and hence proposed a normalisation, known as the Gain Ratio, which is defined by:

C4.5: $GainRatio_A = \frac{I_A}{Info_A}$ *where* $Info_A = \sum_{a \in A} -\frac{\mathrm{N}_a}{\mathrm{N}} log_2 \frac{\mathrm{N}_a}{\mathrm{N}}$

Table 1 Example data set 'Television Repair'

picture quality	sound quality	age	class
poor	good	2	faulty
poor	excellent	1	faulty
good	poor	2	faulty
good	poor	2	faulty
good	excellent	1	not faulty
good	good	1	not faulty
good	good	2	faulty
excellent	good	1	faulty
excellent	excellent	1	not faulty
excellent	good	2	not faulty
good	good	2	faulty
good	good	2	faulty
good	good	1	not faulty
excellent	excellent	1	not faulty
excellent	good	1	not faulty

C4.5 was also developed to include the ability to process numerical data and deal with missing values. Figure 1 presents the tree that result from applying the

ID3 procedure to the examples in Table 1. At each leaf is the class distribution, in the format of [faulty, not faulty].

Once a decision tree has been built, some type of pruning is then usually carried out. Pruning is the term given to that of replacing one or more sub-trees with leaf nodes. There are three main reasons for pruning. One is that it helps to reduce the complexity of a decision tree, which would otherwise make it very difficult to understand [Quinlan 1987], resulting in a faster, possibly less costly classification. Another reason is to help prevent the problem of over-fitting the data.

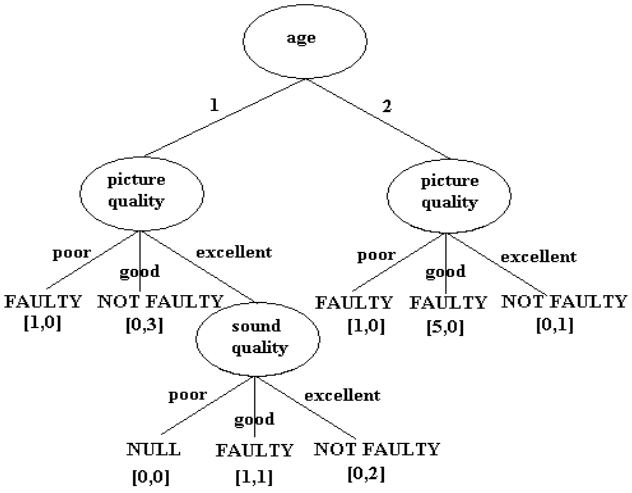


Figure 1 Decision Tree after ID3 has been applied to the data set in Table 1

The third reason is that noisy, sparse or incomplete data sets can cause very complex decision trees, so pruning is a good way to simplify them [Quinlan 1987]. There are several ways to calculate whether a sub-tree should be pruned or not. Quinlan [1987], Knoll et al. [1994] and Bradford et al. [1998a, 1998b] have discussed different methods to do this, for instance, aiming to minimize loss [Bradford et al. 1998a, 1998b], or using misclassification costs to prune a decision tree [Knoll et al. 1994]. This paper focuses on surveying the cost-sensitive tree induction algorithms and readers interested in pruning are referred to the comprehensive review by Frank and Witten [1998].

3. A FRAMEWORK FOR COST-SENSITIVE TREE INDUCTION ALGORITHMS

Section 2 summarized the main idea behind decision tree induction algorithms that aim to maximize accuracy. How can we induce decision trees that minimize costs? The survey reveals several different approaches. First some of the algorithms aim to minimize just costs of misclassification, some aim to minimize just the cost of obtaining the information and others aim to minimize both costs of misclassification as well as costs of obtaining the data. Secondly, the algorithms vary in the approach they adopt. Figure 2 summarizes the main categories that cover all the algorithms found in this survey. There are two major approaches: methods that adopt a greedy approach that aims to induce a single tree, and non-greedy approaches that generate multiple trees. Methods that generate single trees include early algorithms, such as *CS-ID3* [Tan and Schlimmer 1989], that adapt entropy based selection methods to include costs and post-construction methods such as *AUCSplit* [Ferri et al. 2002] that aim to utilize costs after a tree is constructed. Algorithms that utilize non-greedy methods include those that provide a wrapper around existing accuracy based methods, such as *MetaCost*

[Domingos 1999], genetic algorithms, such as *ICET* [Turney 1995], and algorithms that adopt tentative searching methods.

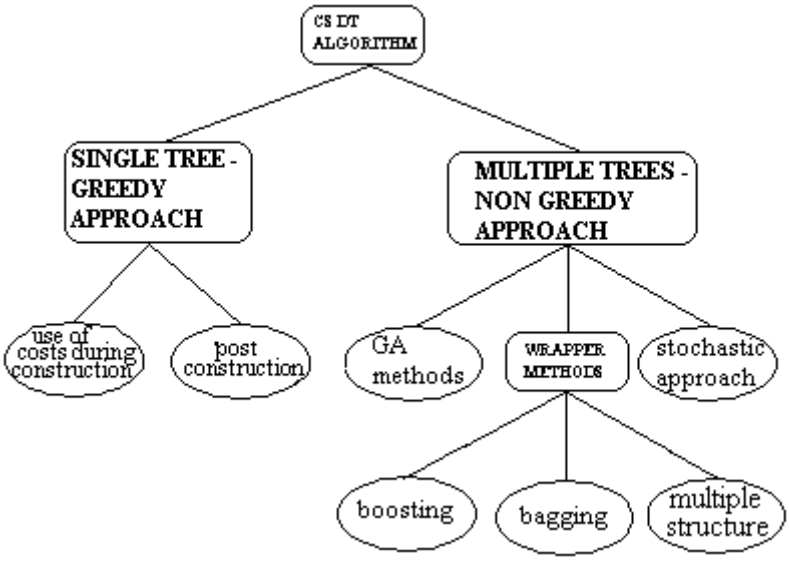


Figure 2 Taxonomy of Cost-Sensitive Decision Tree Induction Algorithms

Table 2 categorizes the algorithms identified in the literature with respect to the taxonomy shown in Figure 2 and shows the significant volume of work in this field in each of the classes. The table also indicates whether the algorithms incorporate test costs, misclassification costs or both. The time line of algorithms, shown as Figure 3, is also interesting. The first mention of the importance of costs dates back to Hunt's [1966] Concept Learning System framework (CLS) that aimed to develop decision trees and recognized that tests and misclassifications could have an economic impact on human decision making. Although, ID3 adopts some of the ideas of CLS, a significant difference in the development was ID3's use of an information theoretic measure for attribute selection [Quinlan 1979]. The use of an information theoretic top-down approach in ID3 influenced much of the early work which focused on methods for adapting existing accuracy based algorithm to take account of costs. These early approaches were evaluated empirically by Pazzani et al. [1994] who observed little difference in performance between algorithms that used cost-based measures and ones that used information gain. This, together with the publication of the results of the *ICET* system [Turney 1995] which used genetic algorithms led to significant interest in developing more novel algorithms, including intense research on the use of boosting and bagging [Ting and Zheng 1998a, 1998b; Ting 2000a, 2000b; Domingos, 1999; Zadrozny 2003a; Lazano and Abe 2008] and more recently, on the use of stochastic approaches [Esmeir and Markovitch 2010, 2011].

The rest of the paper is organized according to the categorization in Table 2, with Section 4 describing the algorithms adopting a single tree, greedy strategy and Section 5 describing the algorithms that use a multiple tree, non-greedy strategy. The Appendix includes a summary of the data sets used by the studies surveyed and which can help identify suitable data for future studies. Table 3 shows the notation and definitions used throughout the paper.

Table 2 Cost-sensitive decision tree induction algorithms categorized with respect to taxonomy by time

ALGORITHM	SOURCE
Use of costs during construction	
GINI Altered Priors	Breiman et al. 1984
CS-ID3 *	Tan & Schlimmer 1989, 1990, Tan 1993
IDX *	Norton 1989
EG2 *	Nunez 1991
LMDT	Draper et al. 1994
Cost-Minimization	Pazzani et al. 1994
C4.5CS	Ting 1998, 2002
EvalCount, MaxCost, AvgCost	Margineantu & Dietterich 2003
Decision Tree with Minimal Costs †	Ling et al. 2004
Decision Tree with Minimal Costs Under Resource Constrains †	Qin et al. 2004
DTNB †	Sheng & Ling 2005
CSNL	Vadera 2005a
LDT	Vadera 2005b
Performance †	Ni et al. 2005
CSGain, CSGainRatio *	Davis et al. 2006
CSTree	Ling et al. 2006a
LazyTree †	Ling et al. 2006b
PM †	Liu 2007
CTS-DT †	Zhang et al. 2007
CS-C4.5 *	Freitas et al. 2007
Post construction	
I-gain (Cost-Laplace prob)	Pazzani et al. 1994
AUCSplit	Ferri et al. 2002
GA methods	
ICET †	Turney 1995
ECCO †	Omielan 2005
CGP	Li et al. 2005
GCT_MC	Kretowski & Grzes 2007
Boosting	
UBoost, Cost-UBoost	Ting & Zheng 1998a
AdaCost	Fan et al. 1999, Ting 2000b
CSB1, CSB2	Ting 2000b
SSTBoost	Merler et al. 2003
GBSE, GBSE-T	Abe et al. 2004
JOUS-Boost	Mease et al. 2007
Lp-CSB, Lp-CSB-PA, Lp-CSB-A	Lozano & Abe 2008
Bagging	
MetaCost	Domingos 1999
MetaCost_A, MetaCost_CSB	Ting 2000a
Cost plus Prior Probability, Cost-Only	Lin & McClean 2000
Costing (Black box)	Zadroznye t al. 2003a, 2003b
B-PET, B-LOT	Moret et al. 2006
Multiple Structure	
Multi-tree	Estruch et al. 2002
Stochastic approaches	
ACT †	Esmeir & Markovitch 2007, 2008
TATA †	Esmeir & Markovitch 2010, 2011

Key: * indicates test costs only incorporated; † indicates both test costs and misclassification costs; otherwise misclassification costs only

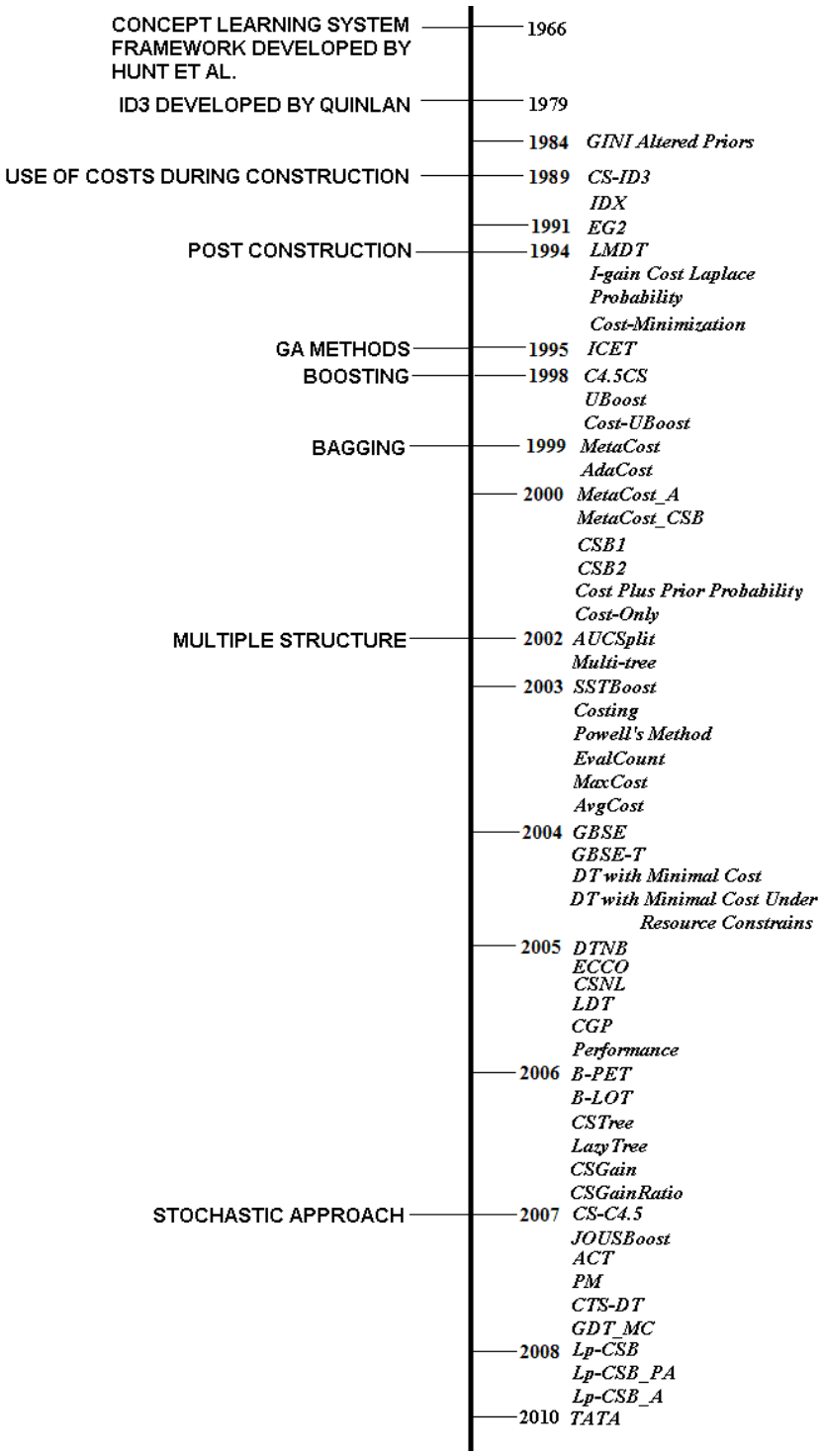


Figure 3 A Timeline of Algorithms

Table 3 Definitions of equations

Symbol	Definition
N	Number of examples in current training set/node
N_i	Number of examples in training set belonging to class i
x	Refers to an example in the training set
node(x)	Leaf node to which the example belongs
k	Number of classes and indicates looping through each class in turn
w	Weights
A	Indicates an attribute
a	Indicates attribute values belonging to an attribute
C_{ij}	Misclassification cost of classifying a class i example as a class j example
C_A	Test cost for attribute A
cost(x,y)	Cost of classifying example x into class y
h_i	The i^{th} hypothesis

4. SINGLE TREE, GREEDY COST-SENSITIVE DECISION TREE INDUCTION ALGORITHMS

As described in Section 2, historically, the earliest tree algorithms developed top-down greedy algorithms for inducing decision trees. The primary advantage of such greedy algorithms is efficiency, though a potential disadvantage is that they may not explore the search space adequately to obtain good results. This section presents a survey of greedy algorithms. The survey identified two major strands of research: Section 4.1 describes algorithms that utilise costs during tree construction and Section 4.2 describes post-construction methods that are useful when costs may change frequently.

4.1 Use of costs during construction

4.1.1. The extension of statistical measures. As outlined in the previous section, top-down decision tree induction algorithms use a measure, such as information gain, to select an attribute upon which the data set will be partitioned during the tree induction process. A reasonable extension, which was taken by a number of early algorithms, was to adapt these information theoretic measures by including costs. These early algorithms retained the top-down induction process and the only differences between them are the selection measures and whether they take account of costs of attributes as well as costs of misclassification.

Five of the algorithms, *CS-ID3* [Tan and Schlimmer 1989], *IDX* [Norton 1989], *EG2* [Nunez 1991] , *CSGain* [Davis et al. 2006] and *CS-C4.5* [Frietas et al. 2007] focus on minimizing the cost of attributes and adapt the information theoretic measure to develop a cost based attribute selection measure, called the Information Cost Function for an attribute A (ICF_A):

EG2: $$ICF_A = 2^{InfoGain_A} - 1/(C_A + 1)^{\omega} \quad (2)$$

CS-ID3: $$ICF_A = (InfoGain_A)^2 / C_A$$

IDX : $$ICF_A = InfoGain_A / C_A$$

CS-C4.5: $$ICF_A = InfoGain_A / (C_A \phi_A)^{\omega}$$

CSGain: $$ICF_A = (N_a/N) * InfoGain_A - \omega * C_A$$

These measures are broadly similar in that they all include the cost of an attribute (C_A) to bias the measure towards selecting attributes that cost less but still take some account of the information gained. The only difference between the measures is the extent of weight given to the cost of an attribute, with *EG2* and *CS-C4.5* adopting a user provided parameter ω that varies the extent of the bias. *CS-C4.5* also includes ϕ_A, a risk factor used to penalize a particular type of

tests, known as delayed tests, which are tests, such as blood tests, where there is a time lag between requesting and receiving the information. The authors of *CSGain* also experiment with a variation, called *CSGainRatio* algorithm where they use the Gain ratio instead of the information gain.

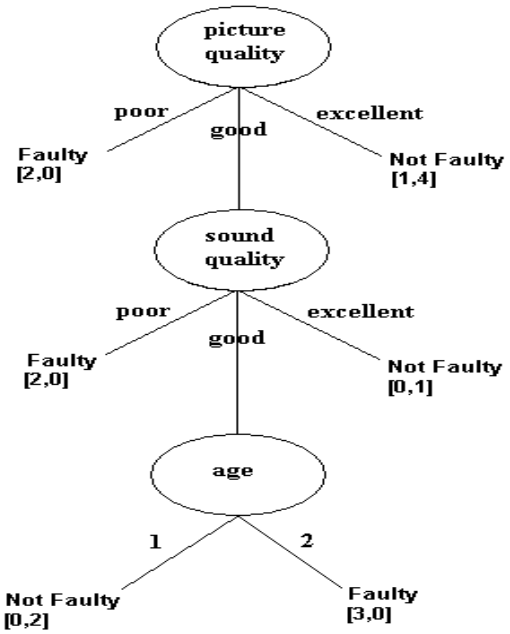


Figure 4 Decision Tree after *EG2* has been applied to the data set in Table 1

Figure 4 presents a cost-sensitive decision tree induced by applying the *EG2* algorithm to the data in Table 1. For illustration purposes, the attributes picture quality, sound quality and age are assigned random test costs of 30, 15 and 1 units respectively. These costs are used in selecting an attribute using the ICF measure resulting in a tree that takes account of the costs of the tests.

Algorithms that continue this adaptation of information theoretic measures but also take account of the misclassification cost as well as the test costs include an approach by Ni et al. [2005], Zhang et al. [2007], Zhang [2010] and Liu [2007]. Although the detailed measures differ, they all aim to capture the trade-off between the cost of acquiring the data and its contribution to reducing misclassification cost. Ni et al. [2005], for example, utilize the following attribute selection measure:

$$Performance:\ ICF_A = ((2^{GainRatio_A} - 1) * DMC_A/(C_A + 1)) * \dot{\omega}_A \qquad (3)$$

where $\dot{\omega}_A$ is the bias of experts for attribute A and DMC_A is the improvement in misclassification cost if the attribute A is used.

As well as using both types of cost, this algorithm makes use of domain experts who assign a value of importance to each of the attributes. If an expert has no knowledge of the importance of an attribute this bias is set to the default value of 1. If some attributes produce the same value for equation (3), preference is given to those attributes with the largest reduction in misclassification costs (DMC_A). If this fails to find an attribute then the attribute with the largest test cost (C_A) is chosen as the aim is to reduce misclassification costs.

Liu [2007] identifies some weaknesses of equation (3), noting that several default values have been used, so develops the *PM* algorithm. Liu [2007] notes that if gain ratios of attributes are small, the values returned by the original algorithm, equation (3), would be small; resulting in the costs of attributes being

ignored. If attributes have large total costs, the information contained in those attributes will be ignored. Other issues are the conflict of applying resource constrains. For instance, the overall aim of this algorithm is to allow for user resource constrains and it is therefore necessary to allow for the fact that users with increased test resources are not concerned as much about the cost of attributes, rather in the reduction of misclassification costs, and alternatively those with limited test resources are more concerned with the cost of the tests in order to reduce the overall costs rather than only reducing the misclassification costs.

In order to trade off between these needs, a solution offered by Liu [2007] is to normalize the gain ratio values and to employ a harmonic mean to weigh between concerns with test costs (low test resources) and reduction in misclassification costs (when test resources are not an issue), additionally a parameter α is used to balance requirements of different test examples with different test resources.

Zhang et al. [2007] take a different approach when adapting the *Performance* algorithm. They focus on the fact that the test costs and misclassification costs are possibly not on the same scale; test costs would be considered on a cost scale of currency whilst misclassification costs, particularly in terms of medical diagnosis, states Zhang et al. [2007], must be a social issue; what monetary value could be assigned for potential loss of life? The adaptation attempts to achieve maximal reduction in misclassification costs from lower test costs. The only difference to equation (3) to produce *CTS (Cost-Time Sensitive Decision Tree)*, is to remove the bias of expert parameter, preferring to address such issues as waiting costs (also referred to in other studies as delayed cost), at the testing stage by developing appropriate test strategies.

The above measures all utilize the information gain as part of a selection measure. An alternative approach, taken by Breiman et al. [1984], is to alter the class probabilities, $P(i)$ used in the information gain measure. That is, instead of estimating $P(i)$ by N_i/N, it is weighted by the relative cost, leading to an altered probability [Breiman, et al. 1984, p114]:

$$\text{Altered Probability}_i = C_{ij} * (N_i/N) \,/\, \sum_j \text{cost}(j)(N_j/N)$$

In general, the cost of misclassifying an example of class *j* may also depend on the class *i* that it is classified into, so Breiman et al. [1984] suggest adopting the sum of costs of misclassification:

$$\text{cost}(j) = \sum_i C_{ij} \qquad (4)$$

Although these altered probabilities can then be used in the Information Gain measure, the method was tried by Pazzani et al. [1994] using the GINI index:

$$\text{Altered GINI} = 1 - \sum_{y=1}^{k} \text{Altered Probability}_y^2$$

C4.5 allows the use of weights for examples, where the weights alter the Information Gain measure by using sums of weights instead of counts of examples. So instead of counting the number of examples with attribute value *a* and class *k*, the weights assigned to these examples would be summed and used in equation (1).

C4.5's use of weights has been utilized to incorporate misclassification costs, by overriding the weight initialization method. For example if the cost to misclassify a faulty example from the example data set in Table 1 is 5, those examples belonging to class 'faulty' could be allocated the weight of 5, and examples belonging to class 'not faulty' could have the weight of 1, so that more weight is given to those examples with the higher misclassification cost. *C4.5CS* is one such algorithm which utilizes this use of weights.

The method of computing initial weights by *C4.5CS* is similar to that of the *GINIAlteredPriors* algorithm developed by Breiman et al. [1984] and Pazzani et al. [1994]. When presented with the same data set, both methods would produce the same decision tree. However Ting [1998] observes that the method which alters the priors would perform poorly as pruning would be carried out in a cost insensitive way, whereas the *C4.5CS* algorithm uses the same weights in its pruning stage. In his experiments with a version which replicates Breiman et al. [1984]'s method, C4.5(π') performs worse that the *C4.5CS* algorithm. He explains this result as owing to different weights in the tree growing stage and the pruning stage.

The sum of all the weights for class *j* in the *C4.5CS* algorithm will be equal to N. The aim of *C4.5CS* is to reduce high cost errors by allocating the highest weights to the most costly errors so that C4.5 concentrates on reducing these errors.

C4.5CS [Ting 1998, 2002]: $weight_j = cost(j)\frac{N}{\sum_i cost(i)N_i}$

where cost(j) and cost(i) are as defined by equation (4).

MaxCost [Margineantu and Dietterich 2003]: $weight_j = max_{1\le i\le k} C_{ji}$

AvgCost [Margineantu and Dietterich 2003]: $weight_j = \frac{\sum_{i=1,i\neq j}^{k} C_{ji}}{(k-1)}$

These latter two algorithms have been designed to solve multi-class problems so the cost matrices involved are not the usual 2 x 2 grids presented when solving two class problems. Instead a k x k matrix is used, the diagonal cells containing the cost of correctly classifying an example, usually zero although for some domains it could well be greater than zero.

Table 4 Example of a cost matrix of a four class problem

Predicted class	**Correct Class**			
	1	**2**	**3**	**4**
1	0	10	2	5
2	100	0	5	2
3	5	2	0	50
4	2	5	25	0

Table 4 presents an example of a cost matrix of a data set where k = 4. The diagonal cells have been assigned zero therefore a correct classification results in zero cost. Two algorithms developed by [Margineantu and Dietterich 2003] use this cost matrix directly to compute initial weights. *MaxCost* uses the worst case cost of misclassifying an example. The maximum value within a column is considered to be the worst case cost of misclassifying an example. For instance, the weight of all class 1 examples will be assigned 100 as that is the maximum misclassification cost in the column corresponding to class 1. *AvgCost* calculates the average cost of misclassifying an example for its weight. Each weight is computed as the mean of the off-diagonal cells in the corresponding column. Using this algorithm, class 1 examples are assigned 35.6. These two algorithms are considered more efficient than others of this type [Margineantu and Dietterich 2003].

Margineantu and Dietterich [2003] also suggest an alternative way of setting the weights, called *EvalCount*, where an accuracy-based decision tree is first induced and then used to obtain the weights. The training data is sub divided into a sub training set and a validation set. The sub training set is then used to grow an accuracy based decision tree. Using this decision tree, the cost of misclassification for each class on the validation set is then measured using the

cost matrix. The weight allocated to a training example is then set to the total cost of misclassifying an example of that class.

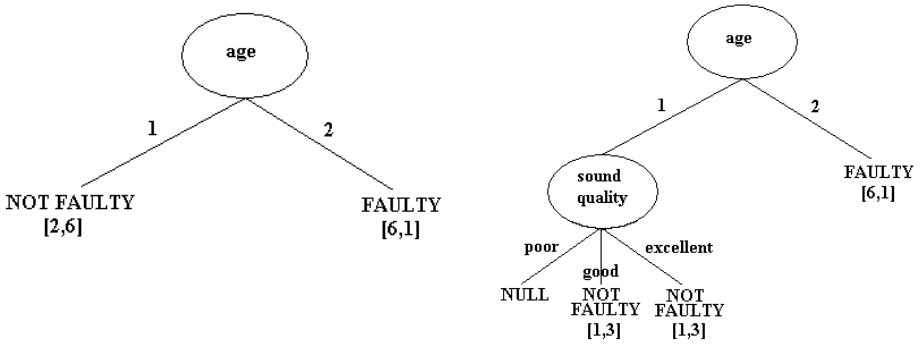
4.1.2 Direct use of costs. Instead of adapting the information gain to include costs, a number of algorithms utilize the cost of misclassification directly as the selection criteria. These algorithms can be subdivided into two groups: those that only use misclassification costs and those which also include test costs.

The central idea with these algorithms is to calculate the expected cost if an attribute is used to divide the examples, compared with the expected cost if there is no further division (i.e. a leaf is assumed). The attribute that results in the most reduction is then selected to divide the examples. Of course, if none of the attributes results in a reduction, then a leaf node is created.

Cost-Minimization [Pazzani et al. 1994], *Decision Trees with Minimal Cost* [Ling et al. 2004] and two adaptations *Decision Trees with Minimal Cost under Resources Constrain* [Qin et al. 2004] and *CSTree* [Ling et al. 2006a] use either misclassification costs or a combination of misclassification costs and test costs to partition the data. *Cost-Minimization*, the simplest of these chooses the attribute which results in the lowest misclassification costs.

One of the main algorithms to use costs directly in order to find the attribute on which to partition data, is *Decision Trees with Minimal Cost* developed by Ling et al. [2004], spawning other adaptations. Expected cost is calculated using both misclassification costs and test costs aiming to minimize the total cost. An attribute with zero or smallest test cost is most likely to be the root of the tree, thus attempting to reduce the total cost. This algorithm has been developed firstly to minimize costs and secondly to deal with missing values in both the training and testing data. In training, examples with missing values remain at the node representing the attribute with missing values. In a study comparing techniques by Zhang et al. [2005], it was concluded that this was the best way to deal with missing values in training examples. How and whether to obtain values during testing are solved by constructing testing strategies and are discussed additionally in Ling et al. [2006b].

To illustrate what happens when only the costs (i.e., no information gain) are used to select attributes, consider the application of the *DT with MC* algorithm to the example in Table 1, where in addition to the test costs we assume the misclassification costs of 50 and 200 for the faulty and not faulty class respectively.



5(a) Tree from DT with MC

5(b) Tree if left branch is expanded.

Figure 5 Decision Tree when *DT with MC* has been applied to data set in Table 1

Figure 5(a) shows the tree induced by DT with MC algorithm, which is very different from the cost-sensitive tree produced by *EG2* (Figure 4*)* and from the tree produced by ID3 (Figure 1). This algorithm employs pre-pruning, that is, it

stops splitting as soon as there is no improvement. Figure 5(b) shows a partial tree obtained, if the left branch was expanded further. The additional attribute that would lead to the least cost is sound quality, with a total cost of 220 units since there are still two faulty examples misclassified but there is the extra cost of 120 units for testing Sound Quality (i.e., 8 examples each costing 15 units). However, the cost without splitting is 100 units (i.e., 2 faulty examples misclassified, with misclassification cost of 50) and hence, in this case, the extra test is not worthwhile.

Ling et al. [2006b] use the algorithm developed in Ling et al. [2004] in a lazy learning framework in order to use different test strategies to obtain missing values on test data and to address problems of delayed tests. Using expected total cost, a tree is induced for each test example using altered test costs, whereby test costs are reduced to zero for examples with known values, thus making them a more desirable choice.

Ling et al. [2004]'s algorithm is further adapted into *CSTree* which does not take into account test costs, using only misclassification costs [Ling et al. 2006a]. *CSTree* deals with two-class problems and estimates the probability of the positive class using the relative cost of both classes and uses this to calculate expected cost.

A different and perhaps more extensive idea is by Qin et al. [2004], who develop an adaptation of the Ling et al. [2004] algorithm *Decision Trees with Minimal Cost under Resource Constrains*. Its purpose is to trade off between target costs (test costs and misclassification costs) and resources. Qin et al. [2004] argue that it is hard to minimize two performance metrics and it is not realistic to minimize both of them at the same time. So they aim to minimize one kind of cost and control the other in a given budget. Each attribute has two costs, test cost and constrain, likewise each type of misclassification has a cost and a constrain value. Both these values are used in the splitting criteria, to produce a target-resource cost decision tree [Qin et al. 2004] and used in tasks involving target cost minimization (test cost) and resources consumption for obtaining missing data.

Decision Tree with Minimal Costs under Resource Constrain:

$$ICF_A = (T - T_A) / Constrain_A$$

$$Constrain_A = (N - o) * r_A + p * C_{ij}(r) + n * C_{ji}(r) + o * C_{ji}(r)$$

where T is the misclassification cost before splitting, T_A is the expected cost if attribute A is chosen, r_A, Cij(r) and Cji(r) are the resource costs for false negatives and false positives respectively, p is the number of positive examples and n the number of negative examples and o the number of examples with missing attribute value.

A different approach than simply using the decision tree produced using direct costs, is suggested by Sheng and Ling [2005], a hybrid cost-sensitive decision tree. They develop a hybrid between decision trees and Naïve Bayes, *DTNB (Decision Tree with Naïve Bayes)*. Decision trees have a structure which is used to collect the best tests but ignores, when classifying, originally known attribute values not appearing in the path taken by a test example. It is argued by Sheng and Ling [2005] that any value is available at a cost, if values are available at the testing stage, these might be useful in order to reduce misclassification costs and to ignore them would be wasting available information. Naïve Bayes can use all known attribute values for classification but has no structure to determine which tests to perform and in what order should they be carried out in order to obtain unknown attribute values. The *DTNB* algorithm aims to combine the advantages of both techniques.

A decision tree is built using expected cost reduction using the sum of test costs and expected misclassification costs to determine whether to further split

the data and on what attribute. Simultaneously a cost-sensitive Naïve Bayes model using Laplace correction and misclassification costs is hidden at all nodes including leaves and is used for classification only of the test examples. The decision tree supplies the sets of tests used in various test strategies and the Naïve Bayes model, built on all the training data, classifies the test examples, thus overcoming problems caused by segmentation of data, that is the reduction of data at lower leaves, and making use of all attributes with known values but which have not been selected during induction so that no information once obtained, is wasted. In experiments, this hybrid method proved to be better in combination than the individual techniques [Sheng and Ling 2005].

4.1.3 Linear and non-linear decision nodes. Most of the early algorithms handle numeric attributes by finding alternative thresholds, resulting in univariate or axis-parallel splits. A number of authors have suggested that this is not sufficiently expressive and adopted more sophisticated multivariate splits. These methods still adopt the top-down decision tree induction process and the primary difference between them, which we summarize below, is whether they adopt linear or non-linear splits and how they obtain the splits.

The *LMDT* algorithm [Draper et al. 1994] was one of the first to go beyond axis-parallel splits. This algorithm aims to develop a decision tree whose nodes consist of Nilsson's [1969] linear machines. A linear machine aims to learn the weights of linear discriminants. Before looking at the *LMDT* algorithm, it is worth understanding the concept of a linear machine, which is central to the *LMDT* algorithm. The following figure summarizes the structure of a linear machine.

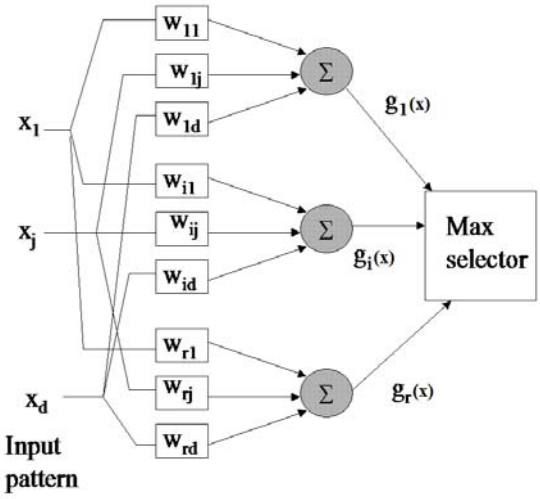


Figure 6 Linear Machine

Each function $g_i(x)$ aims to represent a class i in a winner takes all fashion. A weight w_{ij} represents the coefficient of x_j for the i^{th} linear discriminant function. The training procedure involves presenting an example x that belongs to a class i. If the example is misclassified, say into class j, then the weights of the j^{th} machine can be decreased and the i^{th} machine increased, i.e.:

$$W^i = W^i + c.x$$
$$W^j = W^j - c.x$$

where c is a correction factor, and the W^i and W^j are the weight vectors for the i^{th} and j^{th} linear discriminants.

When the classes are linearly separable, the use of a constant correction rate (i.e. as in a perceptron) is sufficient to determine a suitable discriminant and this

simple procedure converges. However, in general, the classes may not be linearly separable and the above procedure may not converge. Draper et al. [1994] overcame this problem by utilizing a thermal training procedure developed by Frean [1990]. This involved using an annealing parameter β to determine the correction factor *c* as follows:

$$c = \beta^2 / \beta + k \quad \text{where } k = (W_j - W_i)^T x / 2x^T x.$$

where W_j is the weight vector of the i^{th} discriminant function that represents the true class of the example, and W_j is the weight vector of the j^{th} discriminant function that represents the class in which the example is misclassified.

LMDT is altered to make it cost-sensitive by altering its weight learning procedure, with the aim of reducing total misclassification costs. In the modified version, it samples the examples based on the cost of misclassifications made by the current classifier. The training procedure is initialized for each class using a variable 'proportion$_i$', for each class *i*. Next, if the stopping criterion is not met, the thermal training rule trains the linear machine and if the examples have been misclassified, the misclassification cost is used to compute a new value for each 'proportion$_i$'.

An alternative approach to obtaining linear splits, taken in the *LDT* system [Vadera 2005b], is to take advantage of discriminant analysis which enables the identification of linear discriminants of the form [Morrison 1976; Afifi and Clark 1996]:

$$(\mu_1 - \mu_2)\Sigma^{-1}x - \frac{1}{2}(\mu_1 - \mu_2)\Sigma^{-1}(\mu_1 + \mu_2) \leq \ln\left(\frac{C_{21}P(C_2)}{C_{12}P(C_1)}\right) \quad (5)$$

where $\boldsymbol{x}$ is a vector representing the new example to be classified, μ_1, μ_2 are the mean vectors for the two classes, Σ is the pooled co-variance matrix, and $P(C_i)$ is the probability of an example being in class C_i.

Theoretically, it can be shown that equation (5) minimizes the misclassification cost when $\boldsymbol{x}$ has a multivariate normal distribution and when the co-variance matrices for each of the two groups are equal.

This trend of moving towards more expressive divisions is continued in the *CSNL* system [Vadera 2010] that adopts non-linear decision nodes. The approach also utilizes discriminate analysis, and adopts following split that minimizes cost provided the class distributions are multivariate normal:

$$-\frac{1}{2}x^t(\Sigma_1^{-1} - \Sigma_2^{-1})x + (\mu_1^t\Sigma_1^{-1} - \mu_2^t\Sigma_2^{-1})x - k \geq \ln\left(\frac{C_{21}P(C_2)}{C_{12}P(C_1)}\right))$$

$$k = \frac{1}{2}ln\left(\frac{|\Sigma_1|}{|\Sigma_2|}\right) + \frac{1}{2}(\mu_1{}^t\Sigma_1^{-1}\mu_1 - \mu_2{}^t\Sigma_2^{-1}\mu_2) \quad (6)$$

where x is a vector representing the example to be classified, μ_1, μ_2 are the mean vectors for the two classes, $\sum_1$, $\sum_2$ are the covariance matrices for the classes and $\sum_1{}^{-1}$, $\sum_2{}^{-1}$ the inverses of the covariance matrices.

Given that the multivariate assumption may not hold in practice, it may be that utilization of a subset of variables could lead to more cost-effective splits, and hence several strategies for subset selection are explored. One strategy, explored in [Vadera 2005a], is to attempt all possible combinations and select the subset that minimizes cost. However, this strategy is not particularly scalable and results in trees that are difficult to visualize. An alternative strategy, explored in [Vadera 2010], selects two of the most informative features, as measured by information gain, and uses the above equation (6) to obtain non-linear divisions.

4.2 Post construction

If costs are unknown at training time they cannot be used for inducing a tree. Additionally if costs are likely to change, this would mean inducing a tree for every different combination of costs. Hence, various authors have explored how misclassification costs can be applied after a tree has been constructed.

One of the simplest of ways is to change how the label of the leaf of the decision tree is determined. *I-gain Cost-Laplace Probability* [Pazzani et al. 1994] uses a Laplace estimate of the probability of a class given a leaf shown in equation (7). If there are N_i examples of class i at a leaf and k classes then the *Laplace* probability of an example being of class i is:

$$P(i) = \frac{N_i + 1}{k + \sum_{y=1}^{k} N_y} \qquad (7)$$

When considering accuracy only, an example is assigned to the class with the lowest expected error. To incorporate costs, the class which minimizes the expected cost of misclassifying an example into class j is selected, where the expected cost is defined by:

$$Expected\ Cost\ of\ Misclassification\ into\ class\ j = \sum_i C_{ij}\, P(i)$$

Ferri et al. [2002], propose a post construction method based on Receiver Operating Characteristics (ROC) [Swets et al. 2000]. ROC facilitates comparison of alternative classifiers by plotting their true positive rate (on the y axis) against their false positive rate (on the x axis). Figure 7 shows an example ROC, where the true and false rates of four classifiers are plotted. The closer a classifier is to the top left hand corner, the more accurate it is (since the true positive rate is higher and the false positive rate smaller).

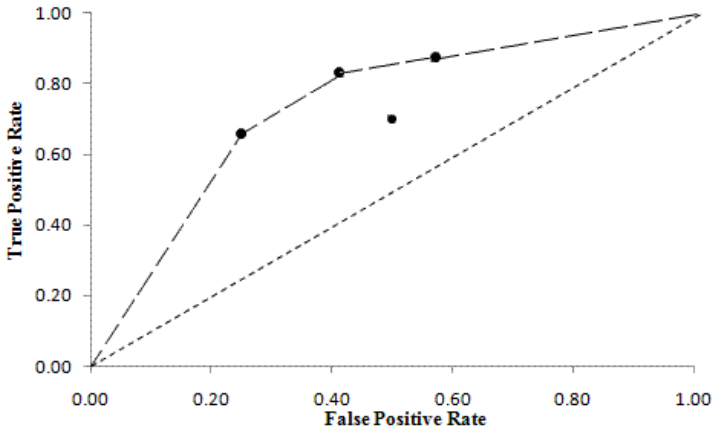


Figure 7 Example ROC

The convex hull created from the points (0,0), the four classifiers and (1,1) represents an optimal front. That is, for any classifier below this convex hull, there is a classifier on the front that is less costly.

The idea behind Ferri et al. [2002]'s approach is to generate the alternative classifiers by considering all possible labellings for the leaf nodes of a tree. For a tree with m leaf nodes, and a two class problem, there are 2^m alternative labels, which could be computationally expensive. However, Ferri et al. [2002] shows that for a two class problem, if the leaves are ordered by the accuracy of one of

the classes, then only m+1 alternative labellings are needed to define the convex hull, where the j[th] node of the i[th] labelling, $L_{i,j}$, is defined by:

$$L_{i,j} = \begin{cases} -ve \; if \; j < i \\ +ve \; if \; j \geq i \end{cases}$$

The convex hull formed by these labellings can then be used to determine the most optimal classifier once the costs of misclassification are known.

5. MULTIPLE TREE, NON-GREEDY METHODS FOR COST-SENSITIVE DECISION TREE INDUCTION

Greedy algorithms have the potential to suffer from local optima, and hence an alternative direction of research has been to develop algorithms that generate and utilize alternative trees. There are three common strands of work: Section 5.1 describes the use of genetic algorithms, Section 5.2 describes methods for boosting and bagging, and Section 5.3 describes the use of stochastic sampling for developing anytime and anycost frameworks.

5.1 Use of Genetic Evolution for Cost-Sensitive Tree Induction

Several authors have proposed the use of genetic algorithms to evolve cost-effective decision trees [Turney 1995]. Just as evolution in nature uses survival of the fittest in order to produce next generations, a pool of decision trees are evaluated using a fitness function, the fittest retained and combined to produce the next generation repeatedly until a cost-effective tree is obtained. This section describes the algorithms that utilize evolution, which vary in the way they represent, generate, and measure the fitness of the trees.

One of the first systems to utilize GAs was Turney's [1995] *ICET* system *(Inexpensive Classification with Expensive Tests. ICET* uses C4.5 but with *EG2's* cost function to produce decision trees, in Section 4.1.

Its populations consists of individuals with the parameters C_{Ai}, ω, and CF, where C_{Ai}, ω are biases utilized in equation (2) and CF is a parameter used by C4.5 for determining the aggressiveness of pruning.

ICET begins by dividing the training set of examples into two random but equal parts: a sub-training set and a sub-testing set. An initial population is created consisting of individuals with random values of C_{Ai}, ω, and CF. C4.5, with the *EG2*'s cost function, is then used to generate a decision tree for each individual. These decision trees are then passed to a fitness function to determine fitness. This is measured by calculating the average cost of classification on the sub-testing set.

The next generation is then obtained by using the roulette wheel selection scheme, which selects individuals with a probability proportional to their fitness. Mutation and crossover are used on the new generation and passed through the whole procedure again. After a fixed number of generations (cycles) the best decision tree is selected. *ICET* uses the GENEtic Search Implementation System (GENESIS, Grefenstette [1990]) with its default parameters including a population size of 50 individuals, 1000 trials and 20 generations.

More recently, Kretowski and Grześ [2007] describe *GDT-MC (Genetic Decision Tree with Misclassification Costs)*, an evolutionary algorithm in which the initial population consists of decision trees that are generated using the usual top down procedure, except that the nodes are obtained using a dipolar algorithm. That is, to determine the test for a node, first two possible examples from the current data set are randomly chosen such that they belong to different classes. A test is then created by randomly selecting an attribute that distinguishes the two examples. Once a tree is constructed, it is pruned using a fitness function. The fitness function used in *GDT-MC* aims to take account of

the expected misclassification cost as well as the size of trees and takes the form [Kretowski and Grześ, 2007]:

$$Fitness\ of\ tree = \left(1 - \frac{EC}{MC}\right)(1 + \gamma . TS)$$

where EC is the misclassification cost per example, MC is the maximal possible cost per example, TS is the number of nodes in the tree and γ is a user provided parameter that determines the extent to which the genetic algorithm should minimize the size of the tree to aid generalization.

The genetic operators are similar in principle to the cross-over and mutation operators, except that they operate on trees. Three cross-over like operators are utilized on two randomly selected nodes from two trees:

- exchange the sub-trees at the two selected nodes.
- if the types of tests allow, then exchange just the tests.
- exchange all sub-trees of the selected nodes, randomly selecting the ones to be exchanged.

The mutation operators adopted allow a number of possible modifications of nodes, including replacing a test with an alternative dipolar test, swapping of a test with a descendent node's test, replacement of a non-leaf node by a leaf node, and development of leaf node into a sub-tree. A linear ranking scheme, coupled with an elitist selection strategy, is utilized to obtain the next generation [Michalewicz 1996].[1]

The *ECCO (Evolutionary Classifier with Cost Optimisation)* system [Omielan 2005] adopts a more direct use of genetic algorithms by mapping decision trees to binary strings and then adopting the standard cross-over and mutation operators over binary strings. Attributes are represented by a fixed size binary string, so for example 8 attributes are coded with 3 bits. Numeric attributes are handled by seeking an axis parallel threshold value that maximizes information gain, thereby resulting in a binary split. The mapping between a tree and its binary string is achieved by assuming a fixed size maximal tree where each node is capable of hosting an attribute which has the most features.[2] Figure 8 illustrates the mapping for a problem where the attributes have two features only. Such a maximal tree is then interpreted by mapping the nodes to attributes, assuming that the branches are ordered in terms of the features. In addition, mutation may result in some nodes with non-existent attributes, which are also translated to decision nodes.

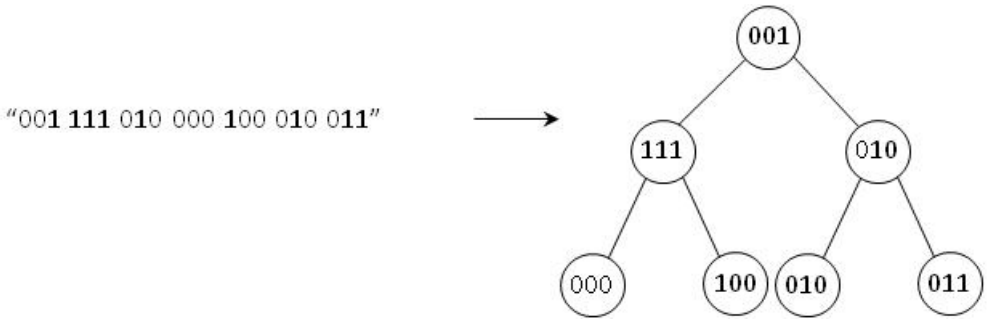


Figure 8 Illustration of mapping

A tree is then populated with the examples in a training set and each leaf node labelled with a class that minimizes the cost of misclassification. A version of the minimums error pruning algorithm that minimizes cost instead of error is used

[1] The elitist strategy ensures that a few of the fittest are copied to the new generation, and the linear ranking strategy ensures some diversity and avoids the fittest don't dominating the evolution to early in the evolution.

[2] The approach works in general for an attribute with more than two features

for pruning. The fitness measure used is the expected cost of classification, taking account of both the cost of misclassification and the cost of the tests. Once genes are mapped to decision trees and pruned, and their fitness obtained, the standard mutation and cross-over operators applied, a new generation of the fittest is evolved and the process repeated a fixed number of cycles. Like *ICET*, *ECCO* adopts the GENESES GA system and adopts its default parameters.

Li et al. [2005] take advantage of the capabilities of Genetic Programming (GP), which enable representation of trees as programs instead of bit strings, to develop a cost-sensitive decision tree induction algorithm. They use the following representation of binary decision trees as programs, defined using BNF [Li et al. 2005]:

<Tree> :: "if-then-else" <Cond><Tree><Tree> | Class
<Cond> :: <Cond> "And" <Cond> | <Cond> "Or" <Cond>
| Not <Cond> | Variable<RelationOperation>Threshold
<RelationOperation> ::= ">" | "<" | "="

Unlike *GDT-MC,* which utilizes specialized mutation and crossover operators, Li et al. [2005] adopt the standard mutation and crossover operators of genetic programming. A tournament selection scheme, in which four individuals are selected randomly with a probability proportional to their fitness, compete to move to the next generation. The fittest of the four is copied to the pool for the next generation and this tournament process repeated to produce the complete mating pool for the next generation. The fitness function employed is also different from *ICET*, *ECCO* and *GDT-MC*. Unlike, these methods, which utilize expected cost, Li et al [2005] propose the following fitness function that is based on the principle that a cost-effective classifier will maximize accuracy (RC) but minimize the false positive rate (RFP):

$$\textit{Constraint Fitness Function} = W_{rc'} * RC - W_{rfp} * RFP$$

Experimentation with this function leads them to the following additional constraint to ensure that accuracy of one of classes is not compromised when the costs of misclassifications are significantly imbalanced:

$$W_{rc} = 1 \text{ if } C_+ \in [P_{min}, P_{max}], 0 \text{ otherwise,}$$

where C_+ is the proportion of examples predicted to be positive, and the P_{min} and P_{max} define the expected range for C_+ that is provided by a user.

5.2 Wrapper Methods for Cost-sensitive Tree Induction

A significant amount of research has been done on accuracy based classifiers, and instead of developing new cost-sensitive classifiers or adapting them as described above, an alternative strategy is to develop wrappers over accuracy based algorithms.

This section describes two approaches for utilizing existing accuracy based algorithms. Section 5.2.1 describes methods based on boosting, where an accuracy based learner is used to generate an improving sequence of hypotheses and Section 5.2.2 describes methods based on bagging that are based on generating and combining independent hypotheses. Section 5.2.3 describes a method which implicitly includes alternative hypotheses but in one structure.

5.2.1 Cost-Sensitive Boosting. Boosting involves creating a number of hypotheses h_t and then combining them to form a more accurate composite hypothesis of the form [Schapire 1999; Meir and Rätsch 2003]:

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \quad (8)$$

where a_t indicates the extent of weight that should be given to $h_t(x)$.

One of the first practical boosting methods, AdaBoost (Adaptive Boosting) works by generating $h_i(x)$ in sequential trials by using a learner on weighted examples that reflect their importance [Freund and Schapire 1996]. It begins by assigning weights of $1/N$ to each example. At the end of each sequential trial, these weights are adjusted so that the weights of misclassified examples are increased, but the weights of correct examples decreased. After a fixed number of cycles, a sequence of trees or hypotheses h_i is available and can be combined to perform classification. The final classification is based on selecting the class that results in the maximum weighted vote as defined by equation (8). There are different versions of AdaBoost with specific weight update rules (e.g., Freund and Schapire [1997], Bauer and Kohavi [1998], Schapire and Singer 1999]). For example, one version that is based on a weak learner capable of producing hypotheses h_t that return a confidence rating in the range [-1,1] uses the following update rule [Schapire 1999]:

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) \quad (9)$$

$$w_{t+1}(x) = \frac{w_t(x)\exp\left(-\alpha_t\, y h_t(x)\right)}{Z_t}$$

where the Z_t is used to normalize the weights so they add up to 1.

Thus, AdaBoost consists of three key steps: the initialization, the weight update equations, and the final weighted combination of the hypotheses. The literature contains a number of algorithms that adapt these three steps of AdaBoost to develop cost-sensitive boosting algorithms.

In particular, Ting and Zheng [1998], which was one of the first studies to utilize boosting for cost-sensitive induction, proposed two adaptations: an algorithm called *UBoost* (*Boosting with Unequal Instance Weights*) and another called *Cost-UBoost* (*UBoost with Cost-Sensitive adaptation*).

UBoost utilizes AdaBoost, except that the weights for each example x, of class j are initialized to the cost of misclassifying an example of class j, and normalized[3] :

$$w_0(x) = cost(j)$$

The cost of misclassifying an example of class i, denoted by *cost(i)* is defined by Ting and Zheng [1998] as in equation (4). Below, we also use the notation *cost(x)* to denote the cost of misclassifying an example x.

In addition, the composite classification rule of equation (8) is adapted to first work out the expected cost of classifying an example $EC_j(x)$, into class j using the combined hypotheses:

$$EC_j(x) = \sum_{t=1}^{T} \alpha_t EC(x, j, h_t)$$

where $EC(x,j,h_t)$ is the expected cost if the example x is classified in class j based on the distribution of examples in the leaf node of the tree h_t that leads to the classification $h_t(x)$.

UBoost then selects the class j that results in the minimum expected cost $EC_j(x)$.

[3] The presentation here assumes that the normalisation of the weights by a factor is Z_t is done at the end of a trial, therefore simplifying the equations.

Ting and Zheng [1998] also propose a method *Cost-UBoost* that extends *UBoost* by also amending the weight update procedure to take account of costs, so that:[3]

$$w_{t+1}(x) = w_t(x).\beta(y', y)$$

where y is the actual class and y' is the predicted class for an example x and ß is defined by:

$$\beta(y', y) = \begin{cases} C_{yy'} \; when \; y' \neq y \\ 1 \qquad when \; y' = y \end{cases}$$

The empirical trials conducted by Ting and Zheng [1998] suggest that *Cost-UBoost* performs better than *UBoost* in terms of minimizing costs of misclassification for two class problems. However, they note that this advantage reduces for multi-class problems and suggest that this is owing to the mapping of different costs of misclassification into a single misclassification cost by equation (4). Later in this section, we describe the more recent work of Abe et al. [2004], and Lozano and Abe [2008]) that develops theoretical foundations for multi-class cost-sensitive boosting problems.

In a follow up study, Ting [2000] propose further variations, named *CSB0*, *CSB1*, *CSB2* and compare their performance to another variation of AdaBoost, known as *AdaCost* [Fan et al. 1999]. *CSB0* is essentially the *Cost-UBoost* algorithm described above, while *CSB1*, *CSB2* and *AdaCost* utilize increasingly sophisticated weight update functions for weak learners that produce the confidence in its prediction $h_t(x) \in [0, 1]$ [Ting, 2000]:

CSB1: $w_{t+1}(x) = w_t(x)\beta(y', y)\exp(-\delta h_t(x))$

CSB2: $w_{t+1}(x) = w_t(x)\beta(y', y)\exp(-\delta h_t(x)\alpha_t)$

AdaCost: $w_{t+1}(x) = w_t(x)\exp(-\delta h_t(x)\alpha_t\beta'(y, y'))$

where δ is -1 if the example is misclassified and +1 if classified correctly, and a α_t is defined as derived in [Shapire and Singer 1999]:

$$\alpha_t = \frac{1}{2} ln\left(\frac{1 + r_t}{1 - r_t}\right)$$

and with r_t defined as follows for the *CSB* family:

$$r_t = \frac{1}{N}\sum_{x \epsilon X} \delta w_t(x) h_t(x)$$

As well as the update equation, the r_t and cost adjustment function ß' are defined differently for *AdaCost*:

$$r_t = \sum_{x \epsilon X} \delta w_t(x) h_t(x) \beta'(y, h(x))$$

$$\beta'(y', y) = \begin{cases} 0.5\, cost(x) + 0.5, & when \; y' \neq y \\ -0.5\, cost(x) + 0.5, & when \; y' = y \end{cases}$$

Ting [2000] evaluates these methods empirically and concludes that the introduction of the α_t in *CSB2* does not lead to a significant improvement and the additional parameters used in *AdaCost* are not particularly effective either.

CSB1 produces more cost-effective results than *AdaCost* in 30 runs while *AdaCost* performs better in 11 runs. Surprisingly, the evaluations also suggest that AdaBoost produces better results than its cost-sensitive version *AdaCost*, which Ting [2000] attributes to the particular definition of ß' that allocates a relatively low reward (penalty) when high cost examples are correctly (incorrectly) classified. This is in contrast to the results presented in [Fan et al. 1999], where *AdaCost* produces better results than AdaBoost when the Ripper learner is used instead of C4.5 as the base learner.

The above adaptations of boosting presume that costs are well-defined in advance. Merler et al. [2003] argue that in medical applications, the costs of false positives or false negatives can only be approximate, and further that during the classification process there two separate phases. In the first phase, the aim is to ensure that the classifier is sensitive and the true positives are maximized whilst the specificity of a classifier is retained within acceptable bounds. In a second phase, a specialist medical consultant would examine the identified positives more carefully, filtering out the false negatives. Hence, for this type of application, they develop a boosting algorithm, *SSTBoost* (*Sensitivity-Specificity tuning Boosting*) that adapts AdaBoost so that the error for the i^{th} example is defined in terms of measures of sensitivity and specificity:

$$\varepsilon_i = (1 - Sensitivity)\pi_{+1}c_{+1} + (1 - Specificity)\pi_{-1}c_{-1}$$

where Π_{+1} Π_{-1} are the class priors and c_{+1}, c_{-1} are the costs of misclassification of the two classes. Sensitivity is the true positive rate and specificity is true negative rate.

With this definition of error, they use equation (9) for α_t:

$$\alpha_t = \left(\frac{1}{2}\right) \ln \left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$

The weight update equation takes the form:

$$w_{t+1}(x) = \begin{cases} w_t(x)\exp\left(-\alpha_t(2 - cost(x))\right), & if example\ x\ is\ classified\ correctly \\ w_t(x)\exp\left(\alpha_t cost(x)\right), & if\ example\ x\ is\ classified\ incorrectly \end{cases}$$

Given specific costs for misclassification, this adaptation of AdaBoost, results in a classifier with a particular sensitivity and specificity. To enable a search for a classifier in a target region of sensitivity and specificity, they relate the costs of misclassifying a positive example, c_+, and cost of misclassifying negative example, c_-, in terms of a single parameter ω:

$$c_{+1} = \omega$$
$$c_{-1} = 2 - \omega$$

This then enables a search over ω by using the method of bisection to find a classifier that aims to be within a user specified region of sensitivity and specificity, meeting their application goals.

The above adaptations of AdaBoost amend the procedure to take account of costs. In contrast, as part of a study that aims to utilize boosting for estimating conditional class probabilities, Mease et al. [2007] describe how AdaBoost can be used directly to develop a procedure called *JOUS-Boost* to perform cost-sensitive boosting. They use a result owed to Elkan [2001] that shows that it is possible to change the distribution of the data to reflect the ratio of costs and such that applying boosting on this changed distribution results in minimization of cost. More specifically, given the cost of misclassification and number of examples of

class 1 and class 2 are N_1, N_2 respectively, the distribution of the data is changed so that the number of examples N_1', N_2' of class 1 and 2, satisfy:

$$\frac{N'_1}{N'_2} = \frac{N_1 C_{12}}{N_2 C_{21}}$$

This change of distribution can be achieved by sampling the original data in a way that results in a smaller data set (under-sampling) or a larger data set (over-sampling). The sampling itself can be done with replacement, where a selected example is returned, or without replacement. Under-sampling can result in loss of data and over-sampling leads to duplication of examples. Mease et al. [2007] carry out experiments on both artificial and real data showing that the duplication due to over-sampling leads to over-fitting when boosting. They then propose a variation, called *JOUS-Boost* (*Over/Under Sampling and Jittering*), that amends the sampling process by adding noise to the features of any duplicated data and provide empirical evidence to show that this helps to reduce over-fitting when AdaBoost is used.

Most of the above algorithms are based on using boosting on two class problems. In two class problems, algorithms such as *UBoost* are able to set the weight of an example in proportion to the cost of misclassifying an example. However, for multi-class problems, an example could be misclassified into several classes, so determining the weight is less obvious. Several authors have proposed methods such as utilizing the sum or average of misclassification into the other classes (e.g. Breiman, et al [1984]; Margineantu [2001]); though as noted above, Ting and Zheng [1998] suggest that use of these methods may explain a reduction in the advantage gained by *Cost-UBoost* over *UBoost* in their empirical evaluations. Abe et al. [2004] also argue that these methods do not have a theoretical basis. Hence, they propose an alternative way of utilizing boosting, called *Gradient Boosting with Stochastic Ensembles* (*GBSE*), for multi-class problems. *GBSE* is motivated by first defining a stochastic hypothesis $H(y \mid x)$ for a class y for an example x based on the individual hypotheses $h_t(x)$ generated by [Abe et al. 2004]:

$$H(y|x) = \frac{1}{T}\sum_{t=1}^{T} I(h_t(x) = y)$$

If $H_t(y \mid x)$ is the composite hypothesis after round t of boosting, then it is formed by combining the previous composite hypothesis H_{t-1} with the new hypothesis, h_t, obtained in round t, weighted by α_t:

$$H_t(y|x) = (1 - \alpha_t t)H_{t-1}(y|x) + \alpha_t I(h_t(x) = y)$$

where $I(E)$ returns 1 if the expression E is true and 0 if E is false, $\alpha = 1/t$, and initially $H_0(x \mid y) = 1/k$, for a k-class problem.

This enables the definition of the expected cost of misclassification over the examples, and using gradient descent, Abe et al. [2004] then derive the following weight update rule, where $w_{x,y}$ is the weight associated with classifying example x in class y:

$$w_{x,y} = cost_{Ht-1}(x) - cost(x, y) \qquad (10)$$

where $cost_{Ht-1}(x)$ is the expected cost of classifying example x by the composite hypothesis $H_{t-1}(x)$.

However, existing boosting methods assume a single weight of importance per example, and not multiple weights, $w_{x,y}$. Hence, to utilize existing boosting methods, there needs to be a mapping to and from multiple weights to single weights per example. Abe et al. [2004] show that this mapping can be achieved by expanding an example $(x, y, (c_1, c_2, \ldots c_k))$, that has features x, class y, and costs of classification into class i defined by c_i, into k examples :

$$S = \{(x, y, max_j c_j - c_i) | i \in (1..k)\} \quad (11)$$

Abe et al. [2004] prove that minimizing cost over this expanded data set is equivalent to minimizing the cost over the original multi-class data. They note that equation (10) can lead to negative weights $w_{x,y}$ which makes it difficult to utilize existing relational weak learners. They therefore transform the examples of equation (11) to the following form:

$$\{((x, y), I(w_{x,y} \geq 0), |w_{x,y}|) \mid x \in X, y \in Y\} \quad (12)$$

A weak learner can be applied to these data and used to induce a relational hypothesis $h_t(x,y)$ and the composite hypothesis revised:

$$H_t(y|x) = (1 - \alpha_t t) H_{t-1}(y|x) + \alpha_t f_t(y|x) \quad (13)$$

where $f_t(y|x)$ converts the relational hypothesis to a stochastic form:

$$f_t(y|x) = \begin{cases} H_{t-1}(y|x), & \text{if no examples } x \text{ such that } h(x,y) = 1 \\ \dfrac{I(h(x,y) = 1)}{|\{y \in Y | h(x,y) = 1\}|}, & \text{otherwise} \end{cases}$$

This formulation defines the mapping needed for *GBSE* to use single weight boosting methods for multi-class problems.

A desirable property of any boosting algorithm is that it should converge and lead to the optimization of its objective. Although this has not been shown for *GBSE*, Abe et al. [2004] show that a variant, called *GBSE-T*, with a fixed α, and the following amendment of the *GBSE* weight update equation (10) converges exponentially:

$$w_{x,y} = \frac{cost_{Ht-1}(x)}{k} - cost(x, y)$$

In a follow up study, Lozano and Abe [2008] develop stronger theoretical foundations for cost-sensitive boosting in which they derive update equations for a family of methods, called *Cost-Sensitive Boosting with p-norm Loss* (*L_p-CSB*) for which they prove convergence. Like the above study, they use stochastic gradient boosting as the methodology, however, instead of aiming to minimize the expected cost of misclassification at each boosting round, they aim to minimize its approximation using the p-norm [Lozano and Abe 2008]:

$$\frac{1}{|S|} \sum_{(x,y,w) \in S} (h(y|x)^p) cost(x, y) \quad for\ p \geq 1$$

where S takes the expanded form defined as equation (11).

They use methodology similar to the derivation of *GBSE* and use gradient descent to show that optimizing this p-norm based objective leads to finding a hypothesis h_t at each round that minimizes [Lozano and Abe 2008, p507]:

$$\sum_{x \in X} \sum_{y \in Y} w_{x,y}(I(h_t(x) = y))$$

where

$$w_{x,y} = H_{t-1}(y|x)^{p-1} cost(x,y)$$

To facilitate the use of a relational weak learner, the examples are translated to the following form, into a similar form to equation (12) in *GSBE* (p508):

$$S = \{((x,y), l, w'_{x,y}) |\ x \in X, y \in Y\}$$

Except that the weights are different from *GSBE* and defined by:

$$\begin{cases} w'_{x,y} = \frac{w_{x,y}}{2} \text{ and } l = 0, & for\ (x,y) when\ y\ does\ not\ coresspond\ to\ the\ minimum\ cost\ class \\ w'_{x,y} = \frac{\sum_{y \in Y'} w_{x,y}}{2} \text{ and } l = 1, & where\ Y' is\ the\ set\ of\ all\ classes\ except\ the\ one\ with\ minimal\ cost \end{cases}$$

Once, a weak learner is applied and a new hypotheses $h_t(x,y)$ obtained, the revised composite hypothesis is defined as in equation (13) with $f_t(x|y) = h_t(x,y)$. Lozano and Abe [2008] prove that this scheme and its related family of schemes are guaranteed to minimize the p-norm based objective, providing a significant theoretical result and understanding of cost-sensitive boosting methods.

5.2.2 Cost-Sensitive Bagging. The main principle of bagging is that producing *n* re-samples of the data set (with replacement), applying a learning procedure to each resample and aggregating the answers leads to better classifiers, particularly for learners that are not stable [Breiman 1996]. This principle is used in *MetaCost* [Domingos 1999], which is one of the earliest systems to utilise cost-sensitive bagging. Thus, *MetaCost* re-samples the data several times and applies a base learner to each sample to generate alternative decision trees. The decisions made on each example by the alternative trees are combined to predict the class of each example that minimizes the cost and the examples relabelled. The relabelled examples are then processed by the base learner, resulting in a cost-sensitive decision tree.

Zadrozny et al. [2003a, 2003b] describe a method called *Costing* that, like *MetaCost* applies a base learner to samples of the data to generate alternative classifiers. However, the sampling process is significantly different from *MetaCost*. Each resample aims to change the distribution of the data so that minimizing error on the changed distribution is equivalent to minimizing cost on the original distribution (i.e., as described for *JOUS-Boost* above). Zadrozy et al. [2003a] argue that using sampling with replacement can lead to overtraining because of the potential for duplication, and sampling without replacement is also problematic since we can no longer assume that the examples selected are independent. Hence, to overcome these shortcomings, Zadrozny [2003a] utilize rejection sampling [Von Neumann 1951] in which an example with cost *c* has a probability of c/Z of being accepted, where Z is chosen as the maximum cost of misclassifying an example. Once these samples, which are proportional to the cost, are generated using rejection sampling, *Costing* applies a base learner to generate *m* classifiers whose outcomes can be aggregated to classify an example. Notice that, unlike *MetaCost*, there is no relabeling of the data in order to generate a single decision tree.

Lin and McLean [2000] develop an approach, in which they use different learners on the same training sample to generate alternative classifiers. As with *MetaCost*, they use the different classifiers to predict the class of each example. However, the labelling of an example x, by a classifier j is based on the risk of classification into two classes:

$$\begin{aligned} \text{Risk}_{1,j} &= \pi_2 * P(2|x,j) * C_{12} \\ \text{Risk}_{2,j} &= \pi_1 * P(1|x,j) * C_{21} \end{aligned} \quad (14)$$

where $\pi 1$, $\pi 2$ are the prior probabilities of the examples in the two classes.

The risk of classification of an example x into a class c is then defined as a weighted sum of the m classifiers:

$$\text{Risk}_{x,c} = \sum_{j=1}^{m} w_j R_{c,j}$$

where w_j, which is the weight associated with classifier j, is the accuracy of the classifier on the training set. The class c that minimizes $\text{Risk}_{x,c}$ is used to label an example x.

Moret et al. [2006] describe a similar method to Lin and McLean [2000], called *Bagged Probability Estimation Trees (B-PETS*), but do not utilize the prior class probabilities, π_i, in equation (14) and also estimate the $P(i|x,j)$ using the distribution of examples in the leaf nodes and Laplace's correction (equation 7) which is known to produce better estimates.

Moret et al. [2006] also propose an alternative way of estimating $P(c|x)$, the probability of an example x being in a particular class c that makes use of lazy option trees. A lazy option tree (LOT) is constructed for an example x, using the usual top-down process except there are two significant differences. First, since the example is known, only tests that are consistent with the example are considered at each node. Secondly, instead of selecting a single best test for each node, the first k best tests are also stored as alternative tests, leading to k sub-trees. An estimate of $P(c|x)$ is then based on the k leaf nodes that the example x falls into. In addition, they also use re-sampling and bagging over lazy option trees (*B-LOT*s), to produce estimates of $P(c|x)$. These estimates of $P(c|x)$ can then be used to select the class that minimizes the risk based on the cost of misclassification.

Given that both *MetaCost* and AdaBoost each result in improved performance, it seems plausible that exploring a combination of the two methods could lead to further improvements. Ting [2000a] investigates this possibility by carrying out an empirical evaluation of two adaptations of *MetaCost*: one, called *MetaCost_A* where the base learner is AdaBoost, and a second, called *MetaCost_CSB*, where the base learner is *CSB0*. The results of the empirical evaluations suggest that there is little to be gained by embedding AdaBoost or *CSB0* within *MetaCos*t. Bagging is known to be particularly effective at reducing variance of due to an unstable base learner [Bauer and Kohavi 1998], which may provide an explanation of why using bagging over AdaBoost or CSB does not result in further improvements.

The comparison in Ting [2000a] also shows that using a cost sensitive base learner for *MetaCost* does result in improvements over a using a cost-insensitive learner, which is also apparent in the empirical results presented in Vadera [2010].

5.2.3 Multiple Structures. Estruch et al. [2002] argue that generating alternative trees such as in boosting and bagging can consume significant space and therefore propose a structure, called *Multi-Tree*, which aims to implicitly include

alternative trees. The central idea is to follow the usual top-down decision induction process, but instead of discarding alternative choices, these are stored as suspended nodes that could be expanded in the future [Ferri-Ramírez 2002; Rissanen 1978]. Figure 9 shows a multi-tree for the example given Table 1 of 'Television Repair' data set. The attributes that have been selected are presented in rectangles, and the suspended nodes, which are in circles, are linked by the dashed lines. The figure also includes the class distribution in each node and is given in the format [number of examples in 'faulty' class, number of examples in 'not faulty' class]. A multi-tree can be expanded to include an additional tree by selecting a suspended node and developing it into a tree using the top-down process but retaining potential attributes as suspended nodes. Estruch et al. [2002] consider alternative methods of selecting which suspended node to expand and adopt a random selection scheme. Thus a multi-tree will implicitly include several trees each of which can be used for classification and whose outcomes can be combined to produce a weighted classification in the same manner to bagging.

Estruch et al. [2002] experiment with different ways of producing this weighted classification by taking advantage of the fact that different decision trees may share the same part of a multi-tree. A multi-tree is not as comprehensible as a single tree, and hence a method for extracting a single tree is developed.

In contrast to MetaCost, where a single tree is obtained by applying a base learner on the re-classified examples, a single tree is extracted by traversing a multi-tree bottom-up, and selecting those suspended nodes that agree the most with the outcomes of the multi-tree using a randomly created data set. They then utilize ROC, as described in Section 4.2, to take account of costs. Estruch et al. [2002] includes an empirical comparison which concludes that it is more efficient in comparison to bagging and boosting. The results for accuracy suggest that bagging produces better results at lower number of iterations while the use of multi-tree produces slightly better results beyond 200 iterations.

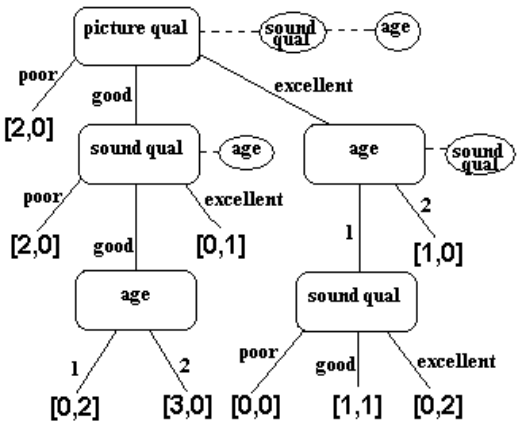


Figure 9 Multi-tree using the example data set

5.3 Stochastic Approach

The greedy methods of induction of trees, described in Section 4, select an attribute after considering its immediate effect on the examples. Several authors have investigated the potential for utilizing a k-lookahead strategy to select attributes by considering their effect deeper down a tree (e.g., Murthy and Salzberg [1995], Dong and Kothari [2001]). That is, for each attribute, a sub-tree of depth k is developed and the attribute that results in the best sub-tree is selected. Although increasing the look-ahead depth k has the potential for increasing the quality of a tree, as Esmeir and Markovitch [2004] point out, this also leads to an exponential increase in the time required for induction.

Hence, in their work they explore the use of stochastic sampling methods to assess the attributes and develop ACT [Esmeir and Markovitch 2007, 2008], a framework for anytime induction of cost-sensitive trees, and TATA [Esmeir and Markovitch,2011], an anycost framework for learning under limited budgets.

ACT (*Anytime Cost-sensitive trees*) [Esmeir and Markovitch 2007, 2008] uses a stochastic tree induction algorithm to generate *r* samples of sub-trees for each value of an attribute. The cost of each of these sub-trees is calculated using the training examples, and the minimum cost utilized as an estimate for the attribute value. The costs of the sub-trees for the attribute values are aggregated to estimate the cost of selecting an attribute, and the minimum cost attribute selected. The first of the *r* samples is generated using the *EG2* algorithm and the remaining samples are generated using a greedy top-down induction process except that the probability of selecting an attribute is proportional to its information cost measure as defined in *EG2* equation (2).

In experiments, *ACT* returned better results than *ICET* (Section 5.1) and *Decision Trees with Minimal Cost* (Section 4.1.2) [Esmeir and Markovitch 2008, p26].

In a more recent study, Esmeir and Markovitch [2011] note that minimising the sum of test and misclassification costs implies that the costs should be on the same scale.[4] They argue that a more realistic goal would be to develop trees that minimise misclassification costs but subject to a constraint that the total cost of the tests utilised is no more than a specified cost.[5] The limit on test costs may be available prior to learning, after learning but before classification, or may be unavailable, leading to algorithms they term as pre-contract, contract and interruptible classifiers. They develop a framework for algorithms for such situations, called *TATA* (Tree classification AT Anycost), that is capable of reducing misclassification costs as the budget for using tests increases.

They develop this framework by first noting that existing top-down tree induction algorithms can be adapted so that the total test cost for any example will be no more than a pre-specified cost. This can be achieved during the tree induction process by only considering those attributes whose cost is below the current available budget, where the current budget is the initial budget less the cost of the attributes used from the root to the current node. Then, they adopt an approach similar to *ACT*, except that the r samples are obtained using an adapted version of C4.5 in which attributes that cost more than the available budget are excluded and attributes are selected stochastically with a probability proportional to their information gain. The samples for each available attribute are used to estimate the misclassification cost and the one with minimal misclassification cost selected. Given a maximum budget available for testing and a suitable sample size r, this achieves the requirements for a pre-contract algorithm.

For a contract algorithm, the budget for test costs in not available until the classification stage. To handle such applications, Esmeir and Markovitch [2011] propose inducing a sequence of trees, t_1, … t_k, which they term a repertoire, with respective budgets $c_1,\ldots,c_k$, where c_1 is set to the cost of the cheapest test, and c_k is set to the maximum cost, where all the tests are used. The number of trees, k, that are used, depends on the amount of time and memory available but also impacts on the time available for the number of stochastic samples, r, that are possible. The k trees could be obtained by discretising the interval from c_1 to c_k into k-1 uniformly spread intervals or in a more sophisticated manner by

[4] As mentioned in section 4.1.1, Zhang et al. [2007], also make the same observation, though they adapt the measure used in the Performance algorithm to represent the trade-off between costs of tests and costs of misclassification.

[5] Greiner et al. (2002) provides some theoretical results for active learning under such budgets.

repeatedly using hill-climbing to subdivide an interval that has the largest gap in terms of expected errors and test cost budgets. [6]

To achieve the goals of an interruptible algorithm, where neither the budgets for learning or the total tests costs are available, they propose developing a repertoire of trees and then to start classification using the tree with the minimum possible budget, and then repeatedly moving on to the tree with the next higher budget until interrupted or reaching the final tree.

An empirical evaluation of *TATA* shows that misclassification costs reduce more rapidly with increasing budget when compared to *EG2*, an adapted version of *EG2* where only attributes within budget are considered and C4.5. The misclassification cost also reduces as the number of stochastic samples, r, increases, with the most significant improvement occurring when one, two or three samples are used, but minimal improvement after three samples.

6. CONCLUSIONS

There has been significant interest in the introduction of costs into decision tree induction. Many ways of introducing costs within the decision tree process have been developed. Whilst there have been accounts of different types of costs, there has been no synthesis of the wide range of studies on cost sensitive algorithms. Hence this paper has carried out an extensive survey of the field with a view to providing an appreciation of the different approaches and algorithms that have been proposed.

A new taxonomy of cost sensitive algorithms has been developed, organizing the algorithms into classes representing the way cost sensitivity has been introduced. The survey revealed two major approaches; greedy, which induces a single tree making decisions with no backtracking and non-greedy, which uses multiple trees and multiple choices to induce trees. Seven classes are defined:

(a) *Use of costs during construction*, whereby attribute selection measures are adapted to include costs. The main differences between the algorithms in this class are the selection measures used and whether costs of tests, misclassification costs or both are incorporated.
(b) *Post construction*, developed when costs are unknown at training time or if the costs are subject to many changes. Differences between these algorithms arise from how the labels for leaf nodes are chosen.
(c) *GA methods*, which utilize evolution, producing populations of decision trees which are evaluated with regard to costs with the fittest being retained and combined. The algorithms vary in the way trees are generated or represented and how the fitness is measured.
(d) *Boosting*, which generates a number of decision trees in sequence using instance weights. The algorithms differ in the way that these weights are initialized, and updated. Other differences between algorithms include how the sampling is done, and how error rates or confidence rates have been calculated in order to give the trees with least error more importance in composite voting methods.
(e) *Bagging*, which generates a number of independent decision trees using re-samples from the training set, thus differing from the trees generated by boosting, being independent of each other similarly to those in the GA methods. Generally these algorithms are wrapping methods, using the decision tree as a sub-routine and wrapping the incorporation of costs around it. Differences between these algorithms are how sampling occurs and in the composite voting method used.

---

[6] More formally, they select an i for which $(E_i - E_{i+1})\ (C_{i+1} - C_i)$ is maximum where the E_i and C_i are the expected error and budgets for the i^{th} tree.

(f) *Multiple structures*, which expands the ideas of generating alternative trees and combining the outcome by having alternative trees in one structure. This shows all possible alternative choices of attribute selection in one decision tree so that alternative choices are not discarded as in the usual decision tree process but are stored and can be expanded in the future.

(g) *Stochastic Approach,* which induces decision trees created by generating *r* stochastic samples of trees rooted at each potential attribute and selecting the attribute that results in the best tree. Varying the number of *r* samples results in the anytime behaviour where quality can improve with more time. As well as anytime behaviour, this approach has been used to produce a framework for anycost behaviour, where misclassification costs reduce as the available total cost for testing increases.

The survey also includes a timeline showing how the field has developed from early algorithms that simply amend selection measures to take account of costs, to the more recent and sophisticated stochastic algorithms that use sampling to induce anycost trees.

Selecting the most appropriate algorithm amongst the many algorithms will depend on various factors including whether an application needs to minimise costs alone, minimise costs of tests and misclassifications, whether there is a fixed budget for test costs, and whether there is a need for anytime or anycost learning. Although, the particular experimental methods, data sets utilised (see Table A.1 in the Appendix) and related systems compared vary, it is possible to form a general view from the empirical evaluations presented in the studies.

A number of the non-greedy algorithms show the benefits of generating multiple trees. Based on the original study by Turney [2005] and the independent comparisons in [Lomax and Vadera, 2011], ICET performs well when aiming to minimise the sum of costs of misclassification and tests, especially when costs of misclassification are uniform[7].

ACT, a system based on stochastic sampling, improves upon the results from ICET for both uniform and non-uniform misclassification costs [Esmeir and Markovitch, 2008].

The use of boosting has developed from the pioneering work on systems such as Cost-UBoost [Ting and Zheng,1998] and AdaCost [Fan et. al, 1999] to JOUS-Boost [Mease et al., 2007] that shows the benefits of adding noise to the sampling process to reduce over-fitting. Lozano and Abe [2008] have advanced our understanding of cost-sensitive boosting by deriving methods such as *Cost-Sensitive Boosting with p-norm Loss* (*L_p-CSB*) that are guaranteed to converge. The recent work of Esmeir and Markovitch [2011] on TATA provides a novel framework for applications where the maximum cost for testing is available in advance, at the classification stage or even later.

Given the relative success of non-greedy algorithms for cost-sensitive tree induction, a fair question is:

> "Is it worth using or even pursuing future research on greedy cost-sensitive decision tree induction algorithms?"

The primary advantage of the greedy algorithms is that they are very efficient and therefore represent a good starting point for applications, and where performance with respect to costs is very good, there may be little benefit in using the more computationally expensive multi-tree methods. Producing similar results to multi-tree methods using single tree methods does represent a major research challenge, but as the work on non-linear decision tree shows [Vadera, 2010], it is possible to produce results comparable to MetaCost and ICET for minimisation of misclassification costs at a fraction of the computational time.

[7] Costs of misclassification are said to be uniform when they are the same for all the classes.

Whether it is possible to extend this to applications that need to take account of costs of tests or budgeted learning remains an open question.

In conclusion, the field of cost-sensitive decision tree learning has a rich and diverse history, providing a strong base for future research that could include: (i) carrying out an independent and comprehensive empirical evaluation that could help method selection based on application requirements (ii) building upon recent advances to develop new algorithms that improve performance or meet new requirements and (iii) developing theoretical foundations that improve our understanding of convergence and the trade-offs between learning time and cost optimisation.

ACKNOWLEDGEMENTS

The authors are grateful to the reviewers and associate editor for their useful comments that have led to improvements to the content and presentation of the paper.

REFERENCES

ABE, N., ZADROZNY, B., LANGFORD, J. 2004. An iterative method for multi-class cost-sensitive learning. *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '04*, 3, August 22-25, Seattle, WA, USA, Kim W., Kohavi R., Gehrke J., DuMouchel W. (Eds).

AFIFI, A. A., CLARK, V. 1996. *Computer-aided multivariate analysis.* 3rd Edition, Chapman & Hall, London.

BAUER, E., KOHAVI, R. 1999. An empirical comparison of voting classification algorithms: bagging, boosting and variants. *Machine Learning Vol 36, Issue 1-2, 105-139.*

BIGGS, D., DE VILLE. B., SUEN, E. 1991. A method of choosing multiway partitions for classification trees. *Journal of Applied Statistics, 18 (1)*, 49-62.

BRADFORD, J. P., KUNZ, C., KOHAVI, R., BRUNK, C., BRODLEY, C. E. 1998a. Pruning Decision Trees with Misclassification Costs. *10th European Conference on Machine Learning (ECML-98),* April 21-23 1998, Chemnitz, Germany, 131-136.

BRADFORD, J. P., KUNZ, C, KOHAVI R, BRUNK, C, BRODLEY, C. E. 1998b. Pruning Decision Trees with Misclassification Costs. *Long version: online at* http://robotics.stanford.edu/~ronnyk/prune-long.ps.gz (Accessed 8 April 2011).

BREIMAN, L., FRIEDMAN J. H., OLSEN R. A., STONE C. J. 1984. *Classification and Regression Trees.* Chapman and Hall/CRC, London.

BREIMAN, L. 1996. Bagging Predictors, *Machine Learning, 24 (2)*, 123-140.

DAVIS, J. V., JUNGWOO, H., ROSSBACH, C. J. 2006. Cost-sensitive decision tree learning for forensic classification. *In Proceedings of 17th European Conference on Machine Learning (ECML),* LNCS 4212, Springer, 622-629.

DOMINGOS, P. 1999. MetaCost: A general method for making classifiers cost-sensitive. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM New York, NY, USA, 155–164.

DONG, M., KOTHARI, R. 2001. Look-ahead based fuzzy decision tree induction. *IEEE Transactions on Fuzzy Systems , Vol 9, No 3*, 461–468.

DRAPER, B. A., BRODLEY, C. E., UTGOFF, P. E. 1994. Goal-Directed Classification using Linear Machine Decision Trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence 16 (9) September*, 888 – 893.

ELKAN, C. 2001. The Foundations of Cost-Sensitive Learning. In *Proceedings of 17th International Joint Conference on Artificial Intelligence (IJCAI'01)*, Morgan Kaufmann, San Francisco, USA, Vol 2, 973 – 978.

ESMEIR, S. AND MARKOVITCH, S. 2004. Lookahead-based algorithms for anytime induction of decision trees. *Twenty-first international conference on Machine learning - ICML '04*, 33, Brodley C.E. (Ed), 257-264.

ESMEIR, S. AND MARKOVITCH, S. 2007. Anytime induction of cost-sensitive trees. In *Proceedings of The 21st Annual Conference on Neural Information Processing Systems (NIPS-2007)*, Vancouver, BC, Canada, 1-8.

ESMEIR, S. AND MARKOVITCH, S. 2008. Anytime Induction of Low-cost, Low-error Classifiers: a Sampling-based Approach. *Journal of Artificial Intelligence Research 33*, 1-31.

ESMEIR, S. AND MARKOVITCH, S. 2010. Anytime Algorithms for Learning Resource-bounded Classifiers. *In Proceedings of the Budgeted Learning Workshop, ICML2010, Haifa, Israel,* June 25.

ESMEIR, S. AND MARKOVITCH, S. 2011. Anytime Learning of Anycost Classifiers. *Machine Learning, Vol 82, No 3,* 445–473.

ESTRUCH, V., FERRI, C., HERNÁNDEZ-ORALLO, J., RAMÍREZ-QUINTANA, M. J. 2002. Re-designing cost-sensitive decision tree learning. In *Workshop de mineria de datos y Aprendizaje.* Iberamia, Seville, November 2002, 33–42.

FAN, W., STOLFO, S. J., ZHANG, J. CHAN, P. K. 1999. AdaCost: misclassification cost-sensitive boosting. *16th International Conference on Machine Learning*, June 27-30 1999, Bled, Slovenia, 97-105.

FERRI, C., FLACH, P., HERNÁNDEZ-ORALLO, J. 2002. Learning decision trees using the area under the ROC curve. In *19th Machine Learning International workshop then conference*, University of New South Wales, Sydney, Australia, 139–146.

FERRI-RAMÍREZ, C., HERNÁNDEZ, J., RAMIREZ, M. J. 2002. Induction of Decision Multi-trees using Levin Search. In *International Conference on Computational Science, ICCS02,* April 21 – 24, Amsterdam, The Netherlands, LNCS, Vol 2329, 166-175.

FRANK, E. AND WITTEN, I. 1998. Reduced-error pruning with significance tests. *Available at* http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.46.2272 (Accessed 8 April 2011).

FREAN, M. 1990. Small Nets and Short Paths: Optimizing Neural Computation. Doctoral thesis, Centre for Cognitive Science, University of Edinburgh.

FREITAS, A., COSTA-PEREIRA, A., BRAZDIL, P. 2007. Cost-Sensitive Decision Trees applied to medical data. *DaWak*, September 3-7, Regensburg, Germany, LNCS 4654, 303-312.

FREUND, Y., SCHAPIRE, R.E. 1996. Experiments with a new boosting algorithm. *13th International Machine Learning workshop then conference*, July 3-6, Bari, Italy, 148-156.

FREUND, Y., SCHAPIRE, R.E. 1997. A decision-theoretic generalization of on-line learning and an application to boosting, *Journal of Computer and System Sciences, 55 (1),* Academic Press, Orlando, Florida, USA, 119-139.

GREINER, R, GROVE, A.J. and ROTH, D. 2002. Learning cost-sensitive active classifiers, *Artificial Intelligence, Vol 139, No 2,* 137-174.

HART, A. E. 1985. Experience in the use of an inductive system in knowledge engineering. In *Research and development in expert systems*. M. A. Bramer (Ed.), Cambridge University Press.

HUNT, E. B., MARIN, J., STONE, P. J. 1966. *Experiments in Induction.* New York, Academic Press.

KNOLL, U., NAKHAEIZADEH, G., TAUSEND, B. 1994. Cost-Sensitive Pruning of Decision Trees. *Proceedings of the 8th European Conference on Machine Learning ECML-94.* Berlin, Germany, Springer-Verlag, 383-386.

KOHAVI, 1996. Scaling up the accuracy of Naïve Bayes Classifier: A Decision Tree hybrid. *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining KDD96*, August 2-4, Portland, Oregon, USA, AAAI Press, 202-209.

KRETOWSKI, M. AND GRZES, M. 2007. Evolutionary induction of decision trees for misclassification cost minimization. *In Proceedings of 8th International Conference on Adaptive and Natural Computing Algorithms,* April 11-14, Warsaw, Poland, ICANNGA, Part 1, LNCS 4431, Springer.

LI, J., LI, X., YAO, X. 2005. Cost-Sensitive Classification with Genetic Programming. *IEEE Congress on Evolutionary Computation*, 2114-2121.

LIN, F. Y. AND MCCLEAN, S. 2000. The Prediction of Financial Distress using a Cost Sensitive approach and Prior Probabilities. *Proceedings of 17th International Conference on Machine Learning, ICML-2000,* June 29-July 2, Stanford University, California, USA.

LING, C. X., YANG, Q., WANG, J., ZHANG, S. 2004. Decision Trees with Minimal Costs. *ACM International Conference Proceeding Series 21st international conference on Machine learning*, Banff, Alberta, Canada, Article No. 69, ISBN: 1-58113-828-5. ACM Press New York, NY, USA.

LING, C. X., SHENG, V. S., BRUCKHAUS, T., MADHAVJI, N. H. 2006a. Maximum profit mining and its application in software development. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '06*, August 20-23, Philadelphia, USA, 929.

LING, C., SHENG, V., YANG, Q. 2006b. Test strategies for cost-sensitive decision trees. *IEEE Transactions on Knowledge and Data Engineering, 18(8)*, 1055-1067.

LIU, X. 2007. A New Cost-Sensitive Decision Tree with Missing Values. *Asian Journal of Information Technology, 6(11)*, 1083–1090.

LOMAX, S., VADERA, S. 2011. An Empirical Comparison of Cost-Sensitive Decision Tree Induction Algorithms. *Expert Systems The Journal of Knowledge Engineering, July, Vol 28, No 3,* 227 – 268.

LOZANO, A.C. AND ABE, N. 2008. Multi-class cost-sensitive boosting with p-norm loss functions. *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '08*, August 24-24, Las Vegas, USA, 506.

MARGINEANTU, D. AND DIETTERICH, T. 2003. A Wrapper Method for Cost-Sensitive Learning via Stratification. Available at http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.27.1102 (accessed 7 April 2011).

MARGINEANTU, D. 2001. Methods for cost-sensitive learning. Doctoral Thesis, Oregon State University.

MEASE, D., WYNER, A. J., BUJA, A. 2007. Boosted Classification Trees and Class Probability/Quantile Estimation. *Journal of Machine Learning Research, 8*, 409-439.

MERLER, S. 2003. Automatic model selection in cost-sensitive boosting. *Information Fusion, 4(1)*, 3-10.

MICHALEVICZ, Z. 1996. *Genetic Algorithms + Data Structures = Evolution Programs.* Third Edition. Springer.

MEIR, R. AND RÄTSCH, G. 2003. An introduction to boosting and leveraging. *Advanced Lectures on Machine Learning,* Mendelson, S., Smola, A. (Eds), Springer, 119-184.

MORET, S., LANGFORD, W. MARGINEANTU, D. 2006. Learning to predict channel stability using biogeomorphic features. *Ecological Modelling, 191(1)*, 47-57.

MORRISON, D. 1976. Multivariate Statistical Methods, *2nd Edition*. McGraw-Hill, New York.

MURTHY, S., KASIF, S., SALZBERG, S. 1994. A System for induction of oblique decision trees. *Journal of Artificial Intelligence Research, 2*, 1-32.

MURTHY, S. AND SALZBERG, S. 1995. Lookahead and pathology in decision tree induction. *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, Montreal, Canada, 1025-1033.

VON NEUMANN, J. 1951. Various techniques used in connection with random digits. Monte Carlo methods. *National Bureau Standards, 12,* 36-38.

Ni, A., ZHANG, S., YANG, S., ZHU, X. 2005. Learning classification rules under multiple costs. *Asian Journal of Information Technology 4*, 1080-1085.

NILSSON, N. J. 1965. Learning Machines, McGraw-Hill, New York.

NORTON, S. W. 1989 Generating Better Decision Trees. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence. IJCAI-89*, August, Detroit, Michigan, USA, 800-805.

NÚNEZ. 1991. The Use of Background Knowledge in Decision Tree Induction. *Machine Learning 6*, Kluwer Academic Publishers, Boston, 231-250.

OMIELAN, A. 2005. Evaluation of a cost sensitive genetic classifier. MPhil thesis, University of Salford.

PAZZANI, M., MERZ, C., MURPHY, P., ALI, K., HUME, T., BRUNK, C. 1994. Reducing misclassification costs. In *Proceedings of the 11th International Conference on Machine Learning.*New Brunswick, New Jersey, USA, 217–225, Available at: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Reducing+misclassification+costs#0 (accessed 7 April 2011).

QIN, Z., ZHANG, S., ZHANG, C. 2004. Cost-sensitive decision trees with multiple cost scales. *Proceedings of the 17th Austrailian Joint Conference on Artificial Intelligence,* December 4-6th Cairns, LNAI 3339, Springer-Verlag, Berlin, G. I. Webb and X Yu (Eds), 380-390.

QUINLAN, J. R. 1979. Discovering rules by induction from large collections of examples. *Expert Systems in the micro electronic age*, D Michie (Ed.), Edinburgh University Press, 168-201.

QUINLAN, J. R. 1983. Learning efficient classification procedures and their application to chess endgames. *Machine Learning: an artificial intelligence approach*, Michalski, Garbonell and Mitchell (Eds.) Tioga Publishing Company, Palo Alto.

QUINLAN, J. R. 1986. Induction of Decision Trees. *Machine Learning 1*, Kluwer Academic Publishers, Boston, 81-106.

QUINLAN, J. R. 1987. Simplifying Decision Trees. *International Journal of Man-Machine Studies, 27*, 221-234.

QUINLAN, J. R. 1993. C4.5: Programs for Machine Learning. Morgan Kaufman, San Mateo, CA.

QUINLAN, J. R., COMPTON, P. J., HORN, K. A., LAZARUS, L. 1987. Inductive Knowledge Acquisition: A Case Study. *Applications of Expert Systems, Chapter 9,* J Ross Quinlan (Ed), Based on the proceedings of the second Australian Conference. Turing Institute Press with Addison-Wesley Publishing Co. ISBN 0-201-17449-9, 137-156.

RISSANEN, J. 1978. Modelling by shortest data description. *Automatica, 14*, 465-471.

SCHAPIRE, R. E., FREUND, Y., BARTLETT, P., LEE, W. S. 1997. Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods. In *Proceedings of the 14th International Conference on Machine Learning*, Nashville, Tennessee, USA, 322-330.

SCHAPIRE, R. AND SINGER, Y. 1998. Improved Boosting Algorithms using Confidence-Rated Predictions. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, July 24-26, Madison, Wisconsin, USA, 80-91.

SCHAPIRE, R. E. 1999. A Brief Introduction to Boosting, In *Proceedings of the 16th International Joint Conference on Artificial Intelligence,* IJCAI99, July 31 – August 6, City Conference Center, Stockholm, Sweden, Vol 2, 1401- 1406.

SCHAPIRE, R. E. AND SINGER, Y. 1999. Improved Boosting Algorithms Using Confidence-Rated Predictions. *Machine Learning 37, (3)*, 297-336.

SHANNON, C. E. 1948. The Mathematical Theory of Communication. *The Bell System Technical Journal, Vol 27*, 379-423.

SHENG, S. AND LING, C. 2005a. Hybrid cost-sensitive decision tree. *9th European Conference on Principles and Practice of Knowledge Discovery in Databases*, October 3-7, Porto, Portugal, LNCS, 3721, 274-284.

SHENG, S., LING, C., YANG, Q. 2005b. Simple Test strategies for cost-sensitive decision trees. *In 16th European Conference on Machine Learning, ECML 2005,* October 3-7, Porto, Portugal, LNCS 3720, 365-376.

SHENG, V. S., LING, C. X., NI, A., ZHANG, S. 2006. Cost-Sensitive Test Strategies. *Proceedings of 21st National Conference of Artificial Intelligence*, July 16-20, Boston, Massachusetts, USA, AAAI Press.

SWETS, J., DAWES, R., MONAHAN, J. 2000. Better Decisions through Science. *Scientific American, October*, 82-87.

TAN, M. AND SCHLIMMER J. 1989. Cost-Sensitive Concept Learning Of Sensor Use in Approach and Recognition. *Proceedings of the 6th International Workshop on Machine Learning. ML-89,* Ithaca, New York, 392-395.

TAN, M. AND SCHLIMMER J. 1990. CSL: A Cost-Sensitive Learning System for Sensing and Grasping Objects. *IEEE International Conference on Robotics and Automation*. Cincinnati, Ohio.

TAN, M. 1993. Cost-Sensitive Learning of Classification Knowledge and Its Applications in Robotics. *Machine Learning, 13*, 7-33.

TING, K. AND ZHENG, Z. 1998a. Boosting cost-sensitive trees. *Proceedings of 1st International Conference on Discovery Science,* Springer-Verlag, London, LNCS*, 1532*, 244–255.

TING, K. M. AND ZHENG, Z. 1998b. Boosting Trees for Cost-Sensitive Classifications. In *Machine Learning: ECML-98 10th European Conference on Machine Learning*, Chemnitz, Germany. Springer, 190-195.

TING, K. 2000a. An empirical study of Metacost using boosting algorithms. *Proceedings of the 11th European Conference on Machine Learning,* Springer-Verlag, London, LNCS 1810, 413–425.

TING, K. 2000b. A comparative study of cost-sensitive boosting algorithms. In *Proceedings of the 17th International Conference on Machine Learning*, June 29 - July 2, Stanford University, San Francisco, USA, 983-990.

TING, K. M. 2002. An Instance-Weighting Method to Induce Cost-Sensitive Decision Trees. *IEEE Transactions on Knowledge and Data Engineering, 14, (3), May/June*, 659-665.

TURNEY, P.D. 1995. Cost-Sensitive Classification: Empirical Evaluation of a Hybrid Genetic Decision Tree Induction Algorithm. *Journal of Artificial Intelligence Research, 2*, 369-409.

TURNEY, P. D. 2000. Types of Cost in Inductive Concept Learning. *Workshop on Cost-Sensitive Learning at the 17th International Conference on Machine Learning (WCSL at ICML-2000)*, Stanford University, California, 15-21.

VADERA, S. 2005a. Inducing cost-sensitive non-linear decision trees. *Technical report. Available at: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Inducing+cost-sensitive+non-linear+decision+trees#0* (Accessed 8 April 2011).

VADERA, S. 2005b. Inducing Safer Oblique Trees without Costs, *Expert Systems: The International Journal of Knowledge Engineering and Neural Networks, Vol 22, No 4*, 206-221.

VADERA, S. 2010. CSNL: A cost-sensitive non-linear decision tree algorithm. *ACM Transactions on Knowledge Discovery from Data (TKDD), Vol 4, Issue 2, May 2010, Article 6*, 1-25.

WEISS, S. M. AND KULIKOWSKI, C. A., 1991. Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning and Expert Systems (Machine Learning Series). Morgan Kaufmann Publishers Inc, California, USA, ISBN 1-55860-065-5.

WINSTON, P. H. 1993. *Artificial Intelligence*. 3rd ed. Addison Wesley, USA, ISBN 0-201-53377-4.

ZADROZNY, B., LANGFORD, J., ABE, N. 2003a. A Simple Method for Cost-Sensitive Learning. *Technical Report RC22666, IBM,2003.* http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.7.7947

ZADROZNY, B., LANGFORD, J., ABE, N. 2003b. Cost-Sensitive Learning by Cost-Proportionate Example Weighting. *Third IEEE International Conference on Data Mining, November 19 - 22, 2003*, Melbourne, Florida, USA, 435.

ZHANG, S., QIN, Z., LING, C., SHENG, S. 2005. "Missing is Useful": Missing Values in Cost-sensitive Decision Trees. *IEEE Transactions on Knowledge and Data Engineering, Vol 17, No 12*, 1689-1693.

ZHANG, S., ZHU, X., ZHANG, J., ZHANG, C. 2007. Cost-Time Sensitive Decision Tree with Missing Values. Knowledge Science, Engineering and Management 4798, 447-459.

ZHANG, S. 2010. Cost-sensitive classification with respect to waiting cost. *Knowledge.-Based Systems, Vol 23, No 5,* 369-378.

APPENDIX

Table A1, given overleaf, shows the top 20 data sets used by the studies in this survey. These data sets have been divided into groups; two class data sets, multi-class data sets and those which have been used as two class and multi-class data sets. The table gives details of how many data sets each study used, the average number of data sets used and how many of the data sets are in the top 20. All data sets in the top 20 are available from the Machine Learning Repository[8]. Some of the studies have used private data sets or those from other sources.

The table also indicates whether the test costs and misclassification costs are provided.

Other data sets which are not in the top 20 listing but are still useful in order to measure performance include: (i) the Soybean data set which has 19 classes, (ii) Thyroid (NN), used by Turney [1995] and others, and is a larger version of the hypothyroid data set and having 3 classes and (iii) Statlog Shuttle, a large data set with 58,000 examples, useful to examine how an algorithm performs with a larger number of examples.

[8] http://archive.ics.uci.edu/ml/index.html

DATASETS			2-Class (from 52)													Both (2)		Multi-class (from 44)					
	Comments	How many out of top 20/Total number of datasets used in experiments	Breast-Wisconsin	Chess (krkp)	Credit Approval	Credit Approval (australia version) statlog	Echocardiogram	Hepatitis ‡	Horse Colic	Congressional Voting Records	Hypothyroid ‡	Mushroom	Pima Indian Diabetes ‡	Tic-tac-toe	Bupa Liver Disorders ‡	Heart Disease (cleveland) ‡	Annealing	Glass	Iris	Solar flare	Splice	Wine
Maximum no of datasets used in a study 28																						
Average no of datasets used in a study 8																						
Number of different studies using the dataset			14	5	5	7	5	12	5	6	5	8	16	10	11	18	6	8	5	7	6	6
ALGORITHM (52)	Studies 39																					
Use of costs during construction			*8*	*2*	*2*	*5*	*1*	*4*	*2*	*1*	*1*	*7*	*7*	*4*	*2*	*9*	*2*	*3*	*1*	-	*2*	-
GINI Altered Priors	Private(1)	3/4										x	x			x						
CS-ID3 *		0/1																				
IDX *		1/1								x												
EG2 *	Private(1)	0/3																				
LMDT	Private(1)	0/1																				
Cost-Minimization	Private(1)	3/4										x	x			x						
C4.5CS		11/22	x		x		x	x	x		x		x			x	x	x			x	
EvalCount, MaxCost, AvgCost		4/6															x	x	x		x	
DT with MC †		3/5	x			x										x						
DT with MC under RC †		3/5	x			x										x						
DTNB †		6/11	x	x		x						x		x		x						
CSNL	Private(1)	8/17	x		x			x	x				x		x	x		x				
LDT	Private(1)	4/6	x					x					x			x						
Performance †		1/1										x										
CSGain, CSGainRatio *		5/7	x			x		x					x		x							
CSTree	Private(1)	0/1																				
LazyTree †		6/10	x	x		x						x		x		x						
PM †		2/2										x		x								
CTS-DT †		2/2										x		x								
CS-C4.5 *	several used only	1/1											x									
Post construction			*1*	*1*				*1*		*1*	*1*	*2*	*2*	*1*	*1*	*1*	-	-	-	-	-	-
I-gain (Cost-Laplace prob)	Private(1)	3/4										x	x			x						
AUCSplit		9/25	x	x				x		x	x	x	x	x	x							
GA methods			*1*	-	-	*1*	-	*2*	-	-	-	-	*3*	-	*3*	*4*	-	*1*	-	-	-	*1*
ICET †		4/5						x					x		x	x						
ECCO †		4/4						x					x		x	x						
CGP	Private(1)	1/3														x						
GCT_MC		7/13	x			x							x		x	x		x				x
Boosting			*3*	*1*	*3*	-	*2*	*2*	*2*	*1*	*3*	-	*2*	*1*	*1*	*2*	*3*	*1*	-	*3*	*3*	-
UBoost, Cost-UBoost		11/22	x		x		x	x	x		x		x			x	x	x			x	
AdaCost	Private(1)	3/7	x		x						x											
CSB1, CSB2		13/14	x	x	x		x	x	x	x	x		x	x	x	x				x		
SSTBoost	Private(1)	0/1																				
GBSE, GBSE-T	KKD99•	3/6															x			x	x	
JOUS-Boost		0/6																				
Lp-CSB, Lp-CSB-PA, Lp-CSB-A	KDD99•	3/9															x			x	x	
Bagging			*2*	*1*	*1*	*1*	*2*	*2*	*2*	*2*	*1*	-	*2*	*1*	*2*	*2*	*2*	*2*	*1*	*2*	*2*	*2*
MetaCost		15/28	x		x		x	x	x	x			x		x	x	x	x	x	x	x	x
MetaCost_A, MetaCost_CSB		17/24	x	x		x	x	x	x	x	x		x	x	x	x	x	x		x	x	x
Cost plus Prior Probability, Cost-Only	Private(1)	0/1																				
Costing (Black box)	KDD98•/ DMEF••	0/2																				
B-PET, B-LOT	Private(1)	0/1																				
Multiple Structure			-	-	-	-	-	-	-	*1*	-	-	-	*1*	-	-	-	-	*1*	-	-	*1*
Multi-tree		4/15								x				x					x			x
Stochastic approaches			-	-	-	-	-	*2*	-	-	-	-	*2*	*2*	*2*	*2*	-	*2*	*2*	*2*	-	*2*
ACT †		9/25						x					x	x	x	x		x	x	x		x
TATA †		9/25						x					x	x	x	x		x	x	x		x

* incorporates test costs

† incorporates test costs + misclassification costs otherwise misclassification costs only

‡ indicates that test costs are available for this dataset, 5 in total

• indicates that misclassification costs are available for this dataset, 5 in total

•• indicates that part misclassification costs are available for this dataset, 1 in total

All top 20 datasets are available from the Machine Learning Repository

The total number of different datasets used is 98

- number of datasets from Machine Learning Repository used 76
- number of datasets from other sources used 22
- artificial (2), KDD repository (2), other researchers (2), DMEF (1), private: supplied from specific domains (10), not specified (5)