Cost-Sensitive Classification with Genetic Programming

Jin Li, Xiaoli Li and Xin Yao
The Centre of Excellence for Research
in Computational Intelligence and Applications (CERCIA)
The School of Computer Science
The University of Birmingham
Edgbaston, Birmingham B15 2TT, UK
{J.Li, X.Li, X.Yao}@cs.bham.ac.uk

Abstract- Cost-sensitive classification is an attractive topic in data mining. Although genetic programming (GP) technique has been applied to general classification, to our knowledge, it has not been exploited to address cost-sensitive classification in the literature, where the costs of misclassification errors are non-uniform. To investigate the applicability of GP to cost-sensitive classification, this paper first reviews the existing methods of cost-sensitive classification in data mining. We then apply GP to address cost-sensitive classification by means of two methods through: a) manipulating training data, and b) modifying the learning algorithm. In particular, a constrained genetic programming (CGP), a GP-based cost-sensitive classifier, has been introduced in this study. CGP is capable of building decision trees to minimize not only the expected number of errors, but also the expected misclassification costs through a novel constraint fitness function. CGP has been tested on the heart disease dataset and the German credit dataset from the UCI repository. Its efficacy with respect to cost has been demonstrated by comparisons with non-cost-sensitive learning methods and cost-sensitive learning methods in terms of the costs.

1 Introduction

In data mining, the task of classification is usually to build a classifier aimed at minimizing the expected number of errors. Typical examples of classifiers are standard top-down decision tree algorithms like CART [5] and C4.5 [27]. Traditionally, such work is based on an assumption that the costs of different misclassification errors are identical. However, in a real world, costs of different misclassification errors are not always same, and some costs are more expensive than others. For example, in medical diagnosis, classifying an illness as a healthy one is a more costly mistake than one of classifying a healthy one as an illness in many cases; for a credit card company, an error of approving a credit card to a bad customer is usually more costly than an error of rejecting a credit card to a good customer. The field of data mining that addresses classification problems with non-uniform costs is known as cost-sensitive classification. The goal of cost-sensitive classification is to build a classifier to minimize the expected misclassification costs rather than to minimize the expected number of misclassification errors.

Cost-sensitive classification is an attractive subject with many applications. Typical examples of applications are medical diagnosis, fraud detection and network troubleshooting. Research work that focuses on building classifiers sensitive only to misclassification costs includes Breiman et al. [5], Chan and Stolfo [6], Provost and Buchanan [26], and Roberts et al. [29], Ting [30], etc. By now, this field has been extended to take into account more other types of costs that could occur in classification such as costs of obtaining attribute values (more details in this regard can be found in Turney [34]). However, this is beyond the scope of this paper. The only costs addressed in this study are costs of misclassification errors. More precisely, here we are merely interested in the cost-sensitive classification problems with non-uniform costs, i.e. different classification errors having unequal costs.

Genetic programming (GP) [16] is part of a more general field known as evolutionary computation inspired by Darwin's evolution theory. In GP, a population (set) of candidate solutions is maintained. For example, a candidate solution could be a decision tree for classification. A fitness function is needed to evaluate the quality of each candidate solution to classify each example. Candidate solutions are selected randomly, biased by their fitness, for involvement in generating members of the next generation. General mechanisms (referred to as genetic operators, e.g. reproduction, crossover, mutation) are used to combine or change the selected candidate solutions to generate offsprings, which will form the population in the next generation. GP has been applied to a broad range of problems with success from traditional optimization in engineering and operational research to non-traditional areas such as composition of music [1] and financial prediction (e.g., [18], [19], [20], [24], [32]). More recently, there is a growing interest in applying GP to data mining and knowledge discovery, which is evidenced by a number of publications [2], [11], [12], [14]. Despite the fact, the potential of data mining with evolutionary algorithms has not been fully explored. To the best of our knowledge, the study of cost-sensitive classification using the technique of GP has not been found in the literature. As a research strategy, it is wise to start with a simple case rather than a complex case. Therefore, this study merely focuses on dealing with two-class cost-sensitive classification problems with unequal misclassification error costs. We leave multiple-class cost-sensitive classification problems to our future research work.

The purpose of this study is twofold. First, we review some existing cost-sensitive learning approaches in machine learning literature and then we would like to evaluate how well the traditional cost-sensitive approaches perform if they were employed by a GP-based classifier. Second, we would like to investigate whether there is a novel approach tailored to the GP-based classifier to address cost-sensitive classification. To achieve both, we also carry out some comparisons between a GP-based classifier and some well-known learning algorithms.

The paper is organized as follows. The next section gives a formal definition of cost-sensitive classification. Section 3 presents the three common approaches in machine learning to cost-sensitive classification: 1) by changing the class distribution of the training data, 2) by modifying the learning algorithms, and 3) by taking the boosting approach. In Section 4, we describe some of details of a GP classifier, and more importantly, we introduce a Constraint Genetic Programming (CGP) with a novel constrained fitness function for approaching cost-sensitive classification. Section 5 discusses the experiments carried out and shows experimental results on two datasets: the heart disease dataset and the German credit dataset from the UCI repository. Finally, discusses on CGP are given and conclusions are drawn in Section 6.

2 Definitions

For a formal setting of the cost-sensitive classification problem, let us start with some notations first. Given a supervised learning problem, a set of labeled examples, or training data (X_i, y) is available to us, where X_i is a vector of attributes (either continuous or nominal values) and y is a class label of X_i, belonging a set of class labels $Y = \{1, 2, ..., J\}$. Also we assume that the training data are drawn from an unknown probability distribution $P(X, y)$. The objective of a learning algorithm is to find a model or a hypothesis h which will be able to map correctly a higher proportion of unlabeled examples drawn from the same distribution. Alternatively, if an incorrect mapping is considered to be a cost (or a loss), the major goal of building a classifier can be treated as minimising the total expected cost or loss ϵ of the hypothesis h:

$$\epsilon(h) = \sum_{(X,y)} P(X, y)C(h(X), y), \tag{1}$$

where $C(h(X), y)$ is the cost function representing the loss incurred by h on an instance $\langle X, y\rangle$. $\epsilon(h)$ is calculated as the sum of the costs or losses of all the individual decisions.

Note that in traditional classification tasks, the loss (or cost) function $C(h(X), y)$ is 1 when $h(X) \neq y$ and 0 otherwise. Therefore, those classifiers are also known as mininsing the expected $0/1$ loss. We will call these classifiers as error-based classifiers (or cost-blind classifiers) later in this paper. Their underlying assumption is that misclassification errors have the same cost. However, typically in cost-sensitive classification tasks, the misclassification error costs are not equal. A cost function needs to be defined.

Without losing generality, a cost function represented as a cost matrix is always assumed to be available before building a classifier. Suppose there is a classification problem with J classes, as well as an associated J x J cost matrix, C. An element $c(i, j)$ of C in row i and column j specifies the cost of misclassifying a class j instance as a class i instance. Please note that the rows in a cost matrix correspond to possible predicted classes, while columns corresponds to actual classes, i.e. row/column = i/j = predicted/actual. If i = j, i.e. the prediction is correct, typically (but not necessarily) the cost is zero.

Cost is domain-specific and can be quantified in different units. For example, cost could be monetary units (i.e. British sterling) for a fraud detection classification; whilst cost could be the severity of an illness in the context of medical diagnosis. Negative cost should be treated as benefit. Following the definitions given by Turney [33], the word “cost” in this paper should be interpreted in its abstract sense.

Mathematically, for a given example X, and if we know conditional probability of each class j $(1 \leq j \leq J)$, $P(j|X)$, the *Bayes optimal* prediction y_{opt} for X should be i, the class that minimises the expected cost (risk) of the labeling decision, i.e.

$$y_{opt} = argmin_{y \in Y} \sum_{j=1}^{J} P(j|X)C(y, j).$$

In other words, assigning X to the class i gives the lowest expected cost among all other possible labeling classes.

Let h be a classifier, and denote $P_h(i, j)$ be the probability that an example, which is selected randomly from the sample distribution, belongs to class j, and is classified by h to be class i. Based on Equation 1, the expected loss of h based on C is

$$L(h) = \sum_{i=1}^{J} \sum_{j=1}^{J} P_h(i, j)C(i, j). \tag{2}$$

Note that $P_h(i, j)$ = $P_h(i|j)P(j)$, where $P(j)$ is the probability that an example belongs to class j, and $P_h(i|j)$ is the conditional probability of classifying an instances as class i given that the instance belongs to class j.

In summary, the general objective of a cost-sensitive classification system is to find such a classifier h that the overall loss based on Equation (2) is minimised.

3 Methods for Cost-Sensitive Classification

As seen in the definitions in Section 2, the introduction of a more complex cost function would certainly make traditional error-based classification methods impotent in solving the cost-sensitive classification problems. This is mainly due to the fact that a more complex cost matrix rather than a simple 0/1 cost function changes the landscape of the loss function $L(h)$ in Equation 2. A classifier which is capable of minimising a simple 0/1 cost function does not guarantee to minimise $L(h)$. To make an error-based classifier cost-sensitive, a common method is to introduce biases into an error-based classification system in three ways: 1) by changing the class distribution of the training data, 2) by modifying the learning algorithm, and 3) by taking the *boosting* approach [13]. In this section, we briefly discuss some traditional existing methods in such three ways. This will shed light on the applicability of GP approach to cost-sensitive classification.

3.1 By Changing the Class Distribution of the Training Data

One of the most common practical approaches to cost-sensitive classification is to change the class distribution of the training data with respect to the cost function and then present an error-based learning algorithm with those modified data. The hope is that the bias introduced on the training data would be able to output a hypothesis that minimises the overall costs of the decisions for unknown future examples. A simple approach is so-called rebalancing (or stratification), i.e. changing the frequency of classes in the training data in proportion to their cost. For simplicity, let us take a two-class classification problem as an example to illustrate the method. A two-class classification always has the following cost matrix C given in Table 1.

	Actual negative	**Actual positive**
Predict negative	C(0, 0) = C_{00}	C(0, 1) = C_{01}
Predict positive	C(1, 0) = C_{10}	C(1, 1) = C_{11}

Table 1: A cost matrix of two-class classification

Here, we take the assumption that the correct classification has no cost, i.e. $C_{00} = C_{11} = 0$; whilst the cost of a false positive is C_{10}, and the cost of a false negative is C_{01}. This assumption is legitimate according to a corollary given in = [21]: *"a cost matrix can always be transformed into an equivalent matrix with zero values on the diagonal"*. The aim of the rebalancing method is to ensure that the proportion of the number of positive examples to the number of negative examples on the training data should be a ratio of C_{01}/C_{10}. This is achievable either by oversampling the examples from the more costly class or by undersampling the examples from the less costly class. For instance, if $C_{01} > C_{11}$, rebalancing can be simply implemented by randomly duplicating some positive examples or by randomly delete some negative examples as many as necessary in accordance with the value C_{10}/C_{01}.

An alternative approach to manipulating the training data is the re-weighting method. The method assigns a weight to each instance of the training data. The size of a weight reflects the influence of misclassifying the case in terms of the cost incurred. An instance is usually assigned a larger weight if it is associated with a higher cost. Whilst the rebalancing method is applicable to any error-based classifiers, the re-weighting method is generally used by the classifiers which can handle instance weights, such as C4.5, Bayesian classifiers, as well as a GP-based classifier proposed in this paper (the way of re-weighting will be elaborated in Section 4).

A more sophisticated method called MetaCost [7] employs a "meta-learning" procedure, i.e. bagging, to relabel the classes of training data and then applies an arbitrary error-based classifier directly to the modified training set to generate a final model.

All these methods make cost-blind classifiers potentially avoid more possible errors with high cost rather than possible errors with low cost by means of modifying the class distribution of training data. This results in reduction of overall cost more likely, but not necessarily as there is a trade-off among errors with different costs. In some circumstance, the gain in reducing high cost errors is outweighed by increasing more low cost errors, resulting in a net increase of cost. Nevertheless, major advantages of these methods are the simplicity without any change in algorithms and the applicability to any existing error-based classification learning algorithms. They do generate better models in terms of overall costs, compared to the models derived on original sample data.

3.2 By Modifying the Learning Algorithms

The second approach to cost-sensitive classification is to alter classification algorithms internally by taking into account the cost function. The approach has been heavily studied with numerous variants. They are implemented either by inducing various biases in the process of building models, or by adjusting thresholds or ordering rules generated. Examples of these are listed as follows.

Roberts et al. [29] applied methods in the process of building decision trees by taking misclassification costs into account either in the class selection criterion at the leaves of the decision tree, or in the test selection criterion at the branches of the decision tree. Ting [30] introduced a simple instance-weighting method which takes cost into account in the process of tree growing and tree pruning. The method effectively converts the standard tree induction procedure that seeks to minimise the number of errors, regardless of cost, to a procedure that seeks to minimise the number of errors with high weight or cost. In particular, when the method is applied to C4.5, the convinced reduction in cost was reported on a representative collection of 12 two-class classification datasets, though the reduction on multiple-class classification is not much. Bradford et al. [4] employed both an extended cost-complexity pruning to loss and a Laplace correction based decision pruning to minimizing loss. Cost-sensitive specialisation [35] involves specializing aspects of a classifier associated with high misclassification costs and generalizing those associated with low misclassification costs, with the aim of reducing the overall costs. In a model called the SBS Testbed, Provost and Buchanan [26] presented a linear function as a bias, which combines the number of correct predictions and the number of costly predictions, to investigate the effect on reduced error costs.

Fawcett and Provost [9] considered non-uniform cost per error in their cellular phone detection task and exhaustively searched (with a fixed increment) for the linear Threshold Unit's threshold that minimizes the total cost. Pazzani et al. [25] presented a method, called RCO (Reduced Cost Ordering) algorithms, which select and order the rules generated by any rule learner such as C4.5, FOCL, to minimize misclassification costs.

Kukar and Kononenko [17] conducted a comparative study of different approaches to cost-sensitive learning with neural networks, including probability estimation and data manipulation techniques. Their results demonstrated that networks trained using a back-propagation version that employed a cost-sensitive weight-update function performed the best.

3.3 By Taking the Boosting Approach

The third approach to cost-sensitive classification is to employ *boosting* method [13]. The algorithm generates a set of different weak classifiers in sequential trials due to each of the training sample data being re-weighted, and then constructs a composite classifier by voting them in terms of the accuracy of each weak classifier. The re-weighting strategy effectively increases the weights of those misclassified instances and decreases the weights of correctly classified instances. The higher the weight, the more the instance influences the weak classifier learned. This causes the learner to focus on those misclassified examples in next trial. Theoretically, it has been proved that a succession of weak classifiers can be boosted to a strong classifier that is not worse, and usually much more accurate than the best weak classifier on the training data. Moreover, the boosting method is applicable to any kinds of base classifiers such as decision trees, Bayesian networks or neural networks. Because of the success of the method in improving accuracy of numerous classifiers([3], [28]), and the fact that it is amenable to cost-sensitive adaption, recent years have witnessed some interesting applications of the boosting method to cost-sensitive classification. Some of them are described below.

AdaCost [8] is a two-class cost-sensitive version of AdaBoost (**Ada**ptive **Boost**ing), an implementation of boosting method [13]. It introduced a misclassification cost adjustment function into the re-writing function of AdaBoost. The function is capable of increasing the weights of costly misclassification instances more aggressively, but decreasing the weights of costly correct classification instances more conservatively. In this way, each weak classifier correctly classifies more expensive examples more likely and the final voted ensemble will also correctly predict more costly instances with the hope of reduction in overall cost .

Making use of the boosting method and based on C4.5 [27], Ting and Zheng [31] developed their cost-sensitive instance-weighting method further into two new methods, namely, UBoost (for **Boost**ing with **U**nequal initial instance weights) and the Cost-UBoost (for **UBoost** with **Cost**-sensitive adaption). UBoost is an ordinary boosting but differs in that it begins with unequal initial instance weights, together with the minimum expected cost criterion in the final classification stage. Cost-UBoost, a variant of UBoost, takes cost into account when inducing weak classifiers, and therefore, it makes the boosting procedure more cost-sensitive. Both methods have been reported to reduce cost significantly.

Cost-sensitive boosting approaches essentially involves both modification of class distribution of training data by means of reweighting and modification of learning algorithms by involving cost while building weak classifiers.

4 Cost-Sensitive GP

Like other classifiers, in theory, a GP-based classifier should also be able to explore these methods discussed above to address cost-sensitive classification. However, due to heavy computational time required by GP and many trials required for the boosting method, employing a GP as a base classifier in boosting is not appropriate, though possible. Therefore, in this study we are only interested in investigating applicability of GP to cost-sensitive classification by changing the class distribution of the training data and by modifying the GP mechanism itself.

To carry out the investigation, first, we develop a GP-based classifier with a simple fitness function, i.e. the *Rate of Correctness* (RC). It aims to achieve a high classification accuracy (equivalent to a low classification error), as much as possible. To apply this error-based GP classifier to cost-sensitive classification, we should manipulate training data using either the rebalancing method or the reweighting method. The reweighting method is easily achievable by taking the weights of instances into the fitness function. In this way, the reweighting method has the same effect as the rebalancing method.

We then consider the second approach to cost-sensitive classification by modifying the GP algorithm. A novel constrained fitness function has been proposed and incorporated into the GP-based classifier. We shall discuss both methods in more detail below.

4.1 A GP-based Classifier

Like other traditional classifiers, such as CART and C4.5, a GP-based classifier should be capable of building decision trees. We achieve this by means of a grammar-based representation, together with an error-based fitness function.

The syntax used in GP to build Genetic Decision Trees (GDTs) can be precisely represented in the *backus normal form* (BNF) grammar, shown in Figure 1. We keep the above grammar as simple as possible simply because any richer one would result in a larger search space for GP. As confirmed by our previous work [19, 20], the syntax that we use here is sufficient to build any similar trees that a standard tree algorithm can create.

The fitness function, one of the most essential components in GP, measures how good an individual is as a solution to the problem at hand. It could take several criteria into account and guide GP to seek the most preferable part of solution space by properly tuning the weights of the criteria involved. Before setting up the fitness function for GP, we need to introduce some measures involved.

As this study focuses on binary classification, some of measures can be defined through a contingency table given in Table 2. The table represents the number of instances belonging to four possible categories after classifying each of examples in a given dataset, i.e. the number of True Positive (TP), the number of True Negative (TN), the number of

$$\langle Tree\rangle ::= \text{“}if{-}then{-}else\text{”}\langle Cond\rangle\langle Tree\rangle\langle Tree\rangle | Class$$

$$\langle Cond\rangle ::= \langle Cond\rangle \text{“}And\text{”}\langle Cond\rangle|$$

$$\langle Cond\rangle \text{“}Or\text{”}\langle Cond\rangle | \text{“}Not\text{”}\langle Cond\rangle|$$

$$Variable\langle RelationOperation\rangle Threshold$$

$$\langle RelationOperation\rangle ::= \text{“} > \text{”}|\text{“} < \text{”}|\text{“} = \text{”}$$

where $Variable$ is an attribute or feature; $Class$ is an integer (1 or 2 for binary classification); $Threshold$ is a real number.

Figure 1: The grammar of GP for constructing trees

	Actual Negative	*Actual Positive*
Predict Negative	**True Negative (TN)($\mathbf{C_{00}=0}$)**	**False Positive (FP)($\mathbf{C_{01}=5}$)**
Predict Positive	**False Negative (FN)($\mathbf{C_{10}=1}$)**	**True Positive (TP)($\mathbf{C_{11}=0}$)**

$$RC = \frac{TP+TN}{TP+FP+TN+FN} \quad RFP = \frac{FP}{FP+TN}$$

Table 2: A contingency table for two-class classification with measures

False Positive (FP), and the number of False Negative (FN). The four figures constitute a matrix, which is referred to as a confusion matrix. Each of instances in the four categories has its corresponding cost associated, denoted by C_{00}, C_{01}, C_{10} and C_{11} respectively (four values in each bracket constitute a cost matrix which is to be used for both experimental datasets in this paper). We defines two measures, namely, the *Rate of Correctness* (RC), and the *Rate of False Positive* (RFP), also given in Table 2. Clearly, as a standard error-based classifier, RC should be taken as a fitness function for the GP classifier aimed at maximising classification accuracy.

To apply this error-based GP classifier to address cost-sensitive classification, we can employ either the rebalancing method or re-weighting method. For GP, the re-weighting approach can be achieved by means of an weighted RC fitness function, denoted by RC_w. RC_w is built by taking into consideration the influence of misclassifying a case in terms of the cost incurred. For example, for the the binary classification given the costs in Table 2, RC_w can be taken as $\frac{TN*C_{10}+TP*C_{01}}{TN*C_{10}+TP*C_{01}+FP*C_{01}+FN*C_{10}} = \frac{TN+TP*C_{01}/C_{10}}{TN+FN+(TP+FP)*C_{01}/C_{10}}$, where all cases belonging to actual positive examples are assigned the weight C_{01}, other cases belonging to actual negative examples are assigned the weight C_{10}. Please note that the second item of formula amounts to measuring the classification accuracy of an error-based classifier over an oversampled dataset. The way of oversampling can be thought as duplicating positive examples with the ratio of C_{01}/C_{10} = 5. The fact illustrates that the rebalancing method can be implemented through the re-weighting method in designing the fitness function for a GP classifier by assigning appropriate weights to different cases in terms of their costs incurred. For this reason, there is no difference between a rebalancing method and a re-weighting method for a GP classifier with the aim of reducing costs.

In Section 5, we shall show the performance of a GP classifier by using this re-weighting method (i.e. taking RC_w as a fitness function) in comparison to the performance of an error-based GP classifier (i.e. taking RC as a fitness function). We shall also compare our GP classifier to the standard decision algorithm C4.5 with and without using the re-weighting method.

4.2 A Novel Constrained Fitness Function

Our intuition to make an error-based GP classifier cost-sensitive by means of modifying learning algorithms is to constitute an appropriate fitness function, which is able to guide GP to search for promising solutions. Ideally, the solutions would be able to trade off between high cost errors and low cost errors so that the overall cost is minimised.

For this sake, we propose a linear fitness function

$$f = w_{rc} * RC - w_{rfp} * RFP,$$

which involves two performance criteria (i.e. RC and RFP), with its corresponding weights. Underlying this function is the hope that resulting GDTs could have lower overall costs by means of maintaining a higher accuracy as much as possible whilst penalising errors incurred with higher costs. Two values of the weights assigned (i.e. w_{rc}, w_{rfp}) reflect the fact to what extent one would like to put emphasis on RC and RFP respectively. Although the idea is valid, we soon realise its drawbacks due to stochastic process of GP. One of the major drawbacks is that resultant GDTs are sensitive to two weights assigned, in particular, the weight w_{rfp}. To overcome the sensitivity of weights assigned, we incorporate a constraint R into the fitness function, f_c, given by.

$$f_c = w_{rc'} * RC - w_{rfp} * RFP, \tag{3}$$

$$where\ w_{rc'} = \begin{cases} 1 & (if\ C_+ \in R = [P_{min}, P_{max}]), \\ 0 & (otherwise), \end{cases}$$

where the range of R is determined by two parameters provided by the user, i.e., P_{min} and P_{max}, which are the expected minimum and maximum of percentage of positive positions predicted respectively in the training data (like most machine learning methods, the assumption is that the test data exhibits similar characteristics); C_+ is actually the percentage of positive positions predicted by a GDT.

The motivation underlying the function f_c is that with the R, given by the user, the error-based GP can be guided to find the GDTs that could avoid more errors with a higher cost, though possibly incur more errors with a lower cost. The hope is that the reduced overall cost of the GDT is achievable, though not necessarily. In f_c, through the conditional weight w_{rc}, the constraint R plays a vital role in finding promising GDTs in terms of the cost. The value of the penalty parameter (i.e., $w_{rc'}$) depends on the fact of whether the GDT being evaluated is able to satisfy the constraint R. If the GDT does, a positive weight, 1, is assigned and therefore a higher positive fitness value is given. Otherwise, zero is set to the weight and therefore a lower negative fitness value is obtained. In this way. only the GDTs are highly rewarded that have a high classification accuracy, a lower rate of false positive, and could avoid making more predictions in the high cost class, rather than in the low cost class.

Setting up an appropriate constraint R, determined by the values of P_{min} and P_{max}, is crucial for success. For example, for the binary classification with the costs given in Table 2, taking $R = [P_{min} = 25\%, P_{max} = 30\%]$, which is less than the actual proportion of positive positions (e.g. 50%), will function generally. However, notably, by taking a much tighter constraint, (e.g. $R = [5\%, 10\%]$), although the function f_c can result in much less high cost errors, it may not work well, as the gain in reducing high cost errors is outweighed by introducing more low cost errors, resulting in a net increase in cost (an extreme case would be the default rule which predicts all cases to be the least expected cost class, where R is $[0\%, 0\%]$).

We refer to the error-based GP classifier with the constraint fitness function f_c, as *Constrained Genetic Programming* (CGP). In Section 5, we shall present the experimental results of CGP, in comparison with some other classification methods.

5 Experiments and Results

To investigate whether an error-based GP classifier and CGP work, we tested both on two of three datasets with costs from a well-known StatLog project [23]: the heart disease dataset and German credit dataset (both are available from the UCI machine learning repository on http://www.ics.uci.edu/ mlearn/MLRepository.html, while the third one, the head injury dataset is not publicly available).

Following [23], we applied the cross-validation method to both datasets. For the heart disease dataset (2 classes, 13 attributes, 270 samples), a 9-fold cross-validation was taken, for the German credit card dataset (2 classes, 24 attributes, 1000 samples), a 10-fold cross-validation was taken. Because of the stochastic property of GP, each GP method have been tested on those folds 10 times each. Therefore, the total 90 and 100 independent runs were carried out for the heart disease dataset and the German credit card dataset respectively. Mean results with corresponding standard deviations over all cycles in cross-validation are reported in this paper for each GP approach.

We measure the performance of a classifier by the average misclassification cost, rather than "the error rate". The average classification cost is computed for any algorithm by multiplying the confusion matrix by the cost matrix, summing the entries and dividing by the total number of test instances. Note that the average cost is equivalent to the error rate if a cost matrix has the unit cost for all errors. In the case of the binary classification here, with a given confusion matrix and a given cost matrix shown in Table 2, the average cost is given by $\frac{FP*C_{01}+FN*C_{10}}{N}$, which ignores both TP and TN simply because both C_{00} and C_{11} are zero. For the heart dataset, an actual positive indicates a heart disease present while an actual negative means a heart disease absent. Likewise, for the German credit card dataset, an actual positive means a bad customer whilst an actual negative indicates a good customer. For both cases, a false positive would incur a higher cost (i.e. $C_{01} = 5$) in contrast to a lower cost for a false negative (i.e. $C_{10} = 1$).

A simple default rule for an error-based classifier is to assign all cases into the most frequent class based on the training data. Similarly, a default rule for cost-sensitive classification is to predict all cases to be the least expected cost class. For example, for the heart dataset, the least expected cost class is the positive class (i.e. "heart disease present"), which produces an lower average cost in comparison to the negative class (i.e. "heart disease absent"). The performance of the default rule could be treated as a bottom line for any algorithm. Any model with a higher average cost compared to the default rule should be considered to be poor.

For brevity, only main parameters running the GP classifier are shown in Table 3. It is worth mentioning that we took the same parameters throughout all experiments in this paper except for some parameters mentioned otherwise.

Population size	3000
Generation	30
Fitness function	RC, RC_w or f_c ($w_{rfp} = 0.6$)
Selection strategy	Tournament selection, size = 4
Max depth of individual program	17
Max depth of initial individual program	4
Crossover rate	0.9
Mutation rate	0.01
Termination criterion	Maximum no. of generations

Table 3: Main parameters for the GP classifier experiments

5.1 Results by the Re-weighting Method

To see whether the traditional re-weighting method works for an error-based GP classifier, we began with assessing the performance of a cost-blind GP classifier using RC. We then assess the performance of the re-weighting method applied to a GP classifier through the experiments using RC_w. Experimental results are shown in Table 4. For comparison, we applied C4.5 to the same datasets as well with and without re-weighting approach and all of results (values in brackets are the standard deviations) reported in Table 4.

Methods	Average Cost	
	Heart Disease	*German Credit*
GP$_{RC}$	0.662 (0.182)	1.125 (0.123)
C4.5	0.744	0.913
GP$_{RC_w}$	0.617 (0.158)	0.566 (0.123)
C4.5$_{re}$	0.467	0.728
Default rule	0.5556	0.700

Table 4: Mean costs of the GP classifier with and without the re-weighting method in comparison with C4.5

Results show that the cost-blind GP and C4.5 do not work well for cost-sensitive classification on both datasets, as mean costs of both methods are even worse than those of the default rules. In contrast, the re-weighting method for both algorithms seems to function to some extent, as C4.5 has a mean cost of 0.467 on the heart disease dataset, and GP has a mean cost of 0.556 on the German credit dataset, both of which are better than the default rules.

5.2 Results by CGP

Methods	Average Cost	
	Heart Disease	*German Credit*
CGP	0.472 (0.145)	0.560 (0.05)
C4.5	0.467	0.728
PART	0.507	0.727
1-KNN	0.561	0.832
2-KNN	0.537	0.818
Naive Bayesian	0.404	0.567

Table 5: Mean costs of the CGP in comparison with C4.5, Part, 1-KNN, 2-KNN and Naive Bayesian using re-weighting method

As mentioned earlier, selecting the range of constraint R is important for CGP to succeed in reducing cost. In experiments, through a trail and error approach, it is found the $R = [30\%, 35\%]$ works very well for both datasets tested here. The question of how to adaptively select a suitable R in the fitness funtion f_c, had better be answered in our future study, as here we focus on studyig whether or not the constraint functions for achieving lower costs.

We show the performances of CGP in Table 5. It achieves a mean cost, 0.472 for the heart disease dataset, and 0.560 for the German credit dataset, both of which are better than the results of the re-weighting method applied on the error-based GP (i.e., 0.472 vs. 0.617; 0.560 vs. 0.566), let alone the default rule. The fact suggests that CGP surpass the re-weighting method in terms of cost. Comparisons with other algorithms would shed light on how good CGP is. We apply three algorithms, namely PART [10](an enhanced rule based method on C4.5), K-Nearest Neighbor (K=1 and K=2) and Naive Bayesian onto both datasets using the re-weighting approach based on the cost matrix given in Table 2. All results of the three methods are also shown in Table 4. In terms of the mean costs produced here, the results demonstrate that CGP has achieved the best performance on the German credit dataset among all, though it is slightly worse than Naive Bayesian (0.472 against 0.404), and C4.5 (0.472 vs. 0.467) on the heart disease.

6 Discussions and Conclusions

To our best knowledge, ICET [33] is the only system that not only takes misclassification costs into account but also involves genetic algorithms. However, unlike CGP, in which genetic programming straightforward plays a main role, ICET uses genetic algorithm as a supplementary means of finding a set of better parameters for a decision tree induction algorithm. The fittest tree is constructed directly through decision tree induction algorithms, rather than genetic algorithms. The novel constrained fitness function that we invent makes it possible for genetic programming to act as a main framework to address cost-sensitive classification problems. Besides, like other cost-sensitive methods in learning systems, ICET cannot provide the mechanism to find varied potential solutions either.

It is worth noting that the constrained fitness function proposed in this paper also has some similarities with the issue in constrained evolutionary optimisation when the penalty method is used (cf. [22]). Both methods transfer a constraint optimisation problem into an unconstraint optimisation problem by introducing a penalty item into the objective function. Both methods highly reward feasible solutions whilst penalizing unfeasible solutions. However, there are some discrepancy between the constraint fitness function and the methods studied in constraint evolutionary optimisation.

The role of the constraint involved is different. The constraints studied in the constrained evolutionary optimisation are solely regarded as constraints that need to satisfy. In contrast, the constraint in the constrained fitness function is not merely a condition. Moreover, it provides a means for balancing two types of errors, i.e. high cost errors and low cost error. The choice of varied constraint R, depends crucially on the cost matrix. We argue that a proper constraint can lead to the reduction in overall cost for cost-sensitive classification.

In this paper, we have reviewed current existing methods of cost-sensitive classification in the literature. As a result, we then have applied GP to address cost-sensitive classification by means of two methods through: a) manipulating training data, and b) modifying the learning algorithm. In particular, CGP has been introduced in this study. CGP is capable of building decision trees to minimize not only the expected number of errors, but also the expected misclassification costs through its novel constraint fitness function. CGP has been tested on the heart disease dataset and the German credit dataset from the UCI repository. Its efficacy with respect to cost has been demonstrated by comparisons with C4.5 and other learning algorithms: PART, K-Nearest Neighbor and Naive Bayesian using the re-weighting method in terms of the costs on both datasets.

Encouraged by the initial results presented in this study on two datasets, we are certainly going to test CGP on more datasets. Another aspect of our future work would be to study the way of setting up the constraint R in the constrained fitness function of CGP. Adaptively dynamic setting up for the constraint R according to various cost matrices and datasets [15], is worth our immediate investigating in the short future.

Acknowledgments

PART, K-Nearest Neighbor and Naive Bayesian used in our experiments are taken from WEKA, which is available at http://www.cs.waikato.ac.nz/ml/weka. Thanks also go to

AWM (Advantage West Midlands) for partial support of this work.

Bibliography

[1] P. Angeline and K. E. Kinnear. *Advances in genetic programming II*. MIT Press, 1996.

[2] W.-H. Au, K. C. C. Chan, and X. Yao. Data mining by evolutionary learning for robust churn prediction in the telecommunications industry. *IEEE Transactions on Evolutionary Computation*, 7 (6):532–545, 2003.

[3] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1-2):105–139, 1999.

[4] J. Bradford, C. Kunz, R. Kohavi, C. Brunk, and C. Brodley. Pruning decision trees with misclassification costs. In *Proc. of the 1998 European Conference on Machine Learning*, pages 131–136, Cairns, 1998. Springer-Verlag.

[5] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, Pacific Grove, CA., 1984.

[6] P. K. Chan and S. J. Stolfo. Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. In *Proc. 4th International Conference on Knowledge Discovery and Data Mining*, pages 164–168, New York, NY, 1998.

[7] P. Domingos. Metacost: A general method for making classifiers cost-sensitive. In *Knowledge Discovery and Data Mining*, pages 155–164, 1999.

[8] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan. AdaCost: misclassification cost-sensitive boosting. In *Proc. 16th International Conf. on Machine Learning*, pages 97–105. Morgan Kaufmann, San Francisco, CA, 1999.

[9] T. Fawcett and F. J. Provost. Adaptive fraud detection. *Data Mining and Knowledge Discovery Vol. 1, Issue 3*, pages 291–316, 1997.

[10] E. Frank and I. H. Witten. Generating accurate rule sets without global optimization. In *Proc. 15th International Conf. on Machine Learning*, pages 144–151. Morgan Kaufmann, San Francisco, CA, 1998.

[11] A. A. Freitas. *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer-Verlag, 2002.

[12] A. A. Freitas. A survey of evolutionary algorithms for data mining and knowledge discovery. In A. Ghosh and S. Tsutsui, editors, *Advances in Evolutionary Computation*, pages 819–845. Springer-Verlag, August 2002.

[13] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, pages 119–139, 1997.

[14] A. Ghosh and A. A. Freitas. (eds.) special issue on data mining and knowledge discovery with evolutionary algorithms. IEEE Trans. on Evolutionary Computation 7(6), December 2003.

[15] J. Joins and C. Houck. On the use of non-stationary penalty functions to solve nonlinear constrained optimisation problems with gas. In H.-P. S. D. F. Z. Michalewics, J.D. Schaffer and H. Kitano, editors, *Proceedings of the First IEEE International Conference on Evolutionary Computation*, pages 579–584. Piscataway, NJ: IEEE Press, 1994.

[16] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[17] M. Kukar and I. Kononenko. Cost-sensitive learning with neural networks. In *European Conference on Artificial Intelligence*, pages 445–449, 1998.

[18] J. Li. *FGP: a Genetic Programming Based Tool for Financial Forecasting*. PhD Thesis, University of Essex, 2001.

[19] J. Li and E. P. K. Tsang. Investment decision making using FGP: A case study. In P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, editors, *Proceedings of the Congress on Evolutionary Computation*, volume 2, pages 1253–1259. IEEE Press, 1999.

[20] J. Li and E. P. K. Tsang. Reducing failures in investment recommendations using genetic programming. In *Computing in Economics and Finance*, Universitat Pompeu Fabra, Barcelona, Spain, 6-8 July 2000.

[21] D. D. Margineantu. *Methods for Cost-Sensitive Learning*. PhD Thesis, Oregon State University, 2001.

[22] Z. Michalewicz and M. Schoenauer. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, Vol. 4, No. 1:1–32, 1996.

[23] D. Michie, D. J. Spiegelhalter, and C. C. Taylor. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, 1994.

[24] M. Oussaidene, B. Chopard, O. Pictet, and M. Tomassini. Practical aspects and experiences - parallel genetic programming and its application to trading model induction. *Journal of Parallel Computing*, 23 No. 8:1183–1198, 1997.

[25] M. Pazzani, C. Merz, P. Murphy, K. Ali, T. Hume, and C. Brunk. Reducing misclassification costs: Knowledge-intensive approaches to learning from noisy data. In *Proc. 11th International Conference on Machine Learning*, pages 217–225, 1994.

[26] F. J. Provost and B. G. Buchanan. Inductive policy: The pragmatics of bias selection. *Machine Learning*, 20 (1/2):35–61, 1995.

[27] J. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., 1993.

[28] J. R. Quinlan. Bagging, boosting, and c4.5. In *Proceedings, Fourteenth National Conference on Artificial Intelligence*, pages 445–449, 1996.

[29] H. Roberts, M. Denby, and K. Totton. Accounting for misclassification costs in decision tree classifiers. In *Proceedings of the International Symposium on Intelligent Data Analysis*, pages 149–156. Baden-Baden, Germany, 1995.

[30] K. M. Ting. An instance-weighting method to induce cost-sensitive trees. *IEEE Transactions on Knowledge and Data Engineering*, 14(3):659–665, 2002.

[31] K. M. Ting and Z. Zheng. Boosting cost-sensitive trees. In *Proceedings of the First International Conference on Discovery Science*, pages 244–255, 1998.

[32] E. P. K. Tsang and J. Li. *EDDIE for Financial Forecasting*. in S-H. Chen (ed.), Genetic Algorithms and Programming in Computational Finance, Kluwer Series in Computational Finance, Chapter 7, 161-174., 2002.

[33] P. D. Turney. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research*, 2:369–409, 1995.

[34] P. D. Turney. Types of cost in inductive concept learning. *Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning (WCSL at ICML-2000)*, 2000.

[35] G. I. Webb. Cost-sensitive specialization. In *Proc. of the 1996 Pacific Rim International Conference on Artificial Intelligence*, pages 23–34, Cairns, 1996. Springer-Verlag.