ECCO: A New Evolutionary Classifier with Cost Optimisation

Adam Omielan, Sunil Vadera

ECCO: A New Evolutionary Classifier with Cost Optimisation

Adam Omielan and Sunil Vadera

School of Computing, Science and Engineering,
University of Salford, Salford M5 4WT, UK

Abstract. Decision tree learning algorithms and their application represent one the major successes of AI. Early research on these algorithms aimed to produce classification trees that were accurate. More recently, there has been recognition that in many applications, aiming to maximize accuracy alone is not adequate since the cost of misclassification may not be symmetric and that obtaining the data for classification may have an associated cost. This has led to significant research on the development of cost-sensitive decision tree induction algorithms. One of the seminal studies in this field has been the use of genetic algorithms to develop an algorithm known as ICET. Empirical trials have shown that ICET produces some of the best results for cost-sensitive decision tree induction. A key feature of ICET is that it uses a pool that consists of genes that represent biases and parameters. These biases and parameters are then passed to a decision tree learner known as EG2 to generate the trees. That is, it does not use a direct encoding of trees. This paper develops a new algorithm called ECCO (Evolutionary Classifier with Cost Optimization) that is based on the hypothesis that a direct representation of trees in a genetic pool leads to improvements over ICET. The paper includes an empirical evaluation of this hypothesis on four data sets and the results show that, in general, ECCO is more cost-sensitive and effective than ICET when test costs and misclassifications costs are considered.

Keywords: Decision Tree Induction, Cost-Sensitive Learning

1 Background

Decision tree learning algorithms have been widely studied since Quinlan first developed the ID3 algorithm [1], with significant impetus given by developments such as C4.5 [2], and its availability in the Weka system [3]. These decision tree learning algorithms aim to take a table of examples as input and produce a decision tree of the kind shown in Figure 1, where there are decision nodes such as Test A, Test B and Test C, and outcomes of the tests decision nodes that are labeled on the links, such as Improve, Same and Deteriorate.

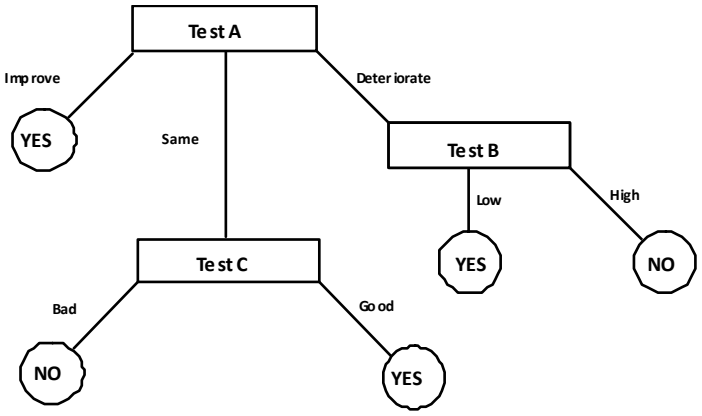

Fig. 1. A decision tree

Early decision tree learning algorithms, such as CART [4], and ID3 [1] aimed to learn such trees from a set of training data. They built decision trees in a greedy fashion, where a test is first chosen as a root. The training examples are then divided into subsets associated with each possible value of the chosen test. The process is then applied recursively until the examples in the subsets all have the similar outcomes, resulting in leaf or decision nodes. A significant step in this greedy algorithm is the criteria for selecting the next test. Most of the early algorithms adopted information theoretic measures that selected tests on the basis of the amount of information gained towards the final classification. The primary aim of these greedy algorithms was to produce accurate decision trees, where accuracy was estimated by the proportion of cases for which the data in a testing set were correctly classified.

However, several authors recognized that maximizing accuracy is not adequate for many real world applications and that costs need to be taken into account (e.g., [4,5,6]). There are several types of costs but the main ones include the costs of misclassifying an example and the cost of acquiring information [7]. For example, in a medical application misclassifying a person as healthy when they are ill can be higher than misclassifying them as ill, and the cost of carrying out an MRI scan can be higher than a blood test.

Research on development of cost-sensitive decision tree algorithms can be broadly classified into the following main categories:

1. Algorithms that adopt the Greedy algorithm but that adapt the information theoretic selection measures to include costs. The key difference amongst algorithms in this category is how the information gain measure is adapted to include costs and whether they take account of just the costs of the tests, or also take cost of misclassification. Algorithms that take account of just costs of attributes include CS-ID3 [8], IDX [9], EG2 [10] , CSGain [11]. Algorithms that also adapt the information

theoretic measure to include costs of misclassification include PM [12], and CS-4.5 [13].

2. Algorithms that utilise bagging and boosting methods that generate alternative trees and combine them in a way that reduces cost. For example, the MetaCost system [14] resamples the data several times and applies a base learner (such as C4.5) 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 relabeled. The relabeled examples are then processed by the base learner, resulting in a cost-sensitive decision tree. Other examples of systems that adopt bagging include B-PET & B-LOT [15], and examples of systems that use boosting methods include AdaCost [16] and Lp-CSB [17].
3. The use of Genetic Algorithms (GAs) to generate and evolve cost-sensitive trees. The idea with these methods is to begin with a genetic pool, select the fittest, apply evolution operators such as mutation and crossover to generate a new pool and repeat the evolution cycles, resulting in improved pools. The main algorithm in this category is Turney's ICET [5] (Inexpensive Classification with Expensive Tests). 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 CAi, ω, and CF, which are parameters required by C4.5 and EG2 . 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 the best decision tree is selected.

A comprehensive survey of these algorithms that includes a framework and timeline covering over 50 algorithms can be found in Lomax and Vadera [18]. This paper focuses on this last category: algorithms that use genetic algorithms for generating cost-sensitive decision tree algorithms. By far the most widely known and cited algorithm in this category is Turney's ICET [5]. In our previous empirical evaluations, ICET produced some of the best results in comparison to algorithms from other categories [19]. As mentioned above, a key feature of ICET is that the population of individuals consists of biases that are utilized when C4.5 is used to generate decision trees. That is the individuals in the population are not direct encodings of trees.

This paper presents a new algorithm called ECCO that is based on the hypothesis that direct encoding of trees in a pool could improve upon the results of Turney's seminal system ICET.

The next section of the paper describes how ECCO is developed using a GA and is followed by a section that presents an empirical evaluation of the hypothesis that a direct representation of trees leads to improvements over the ICET system.

2 The ECCO algorithm

To develop a cost-sensitive decision tree algorithm that is based on the use of a genetic algorithm we need to address the following questions:

- How can the trees be represented as genes?
- What mutation and cross-over operators are appropriate?
- What sort of fitness function is appropriate?

The following subsections describe how these questions are addressed, leading to the ECCO algorithm.

2.1 Encoding the tests and Trees as Genes

The genes in GAs are represented as bit strings. To represent a tree as a bit string, we need to first code each test. Each test is given a unique identification number, from 1 to n, where n is the total number of tests. Each identifier is then converted to its binary equivalent. To code a tree as a fixed length binary string, we need to assume a fixed size tree. The maximal size of a tree is determined by the number of possible values of the tests. For example, a test such as 'It has wings' has only two outcomes, true or false. More complex tests give rise to more outcomes, e.g. A test such as 'How many eggs?' when classifying a recipe may well have '0, 1, 2, 3' eggs possible, and so four child nodes are needed. Given that we know the tests and the maximal number of possible outcomes of tests, we can compute the maximal size of a tree. This can then be used to compute the length of the bit string required to represent a tree. Figure 2 illustrates a mapping for tree of depth 3 assuming a maximal of two outcomes per test (the idea extends to n-ary trees).

It's worth noting that continuous variables are handled by using the process used in systems such as C4.5 where they discretized to axis parallel binary tests (See Quinlan C4.5 [2] for details).

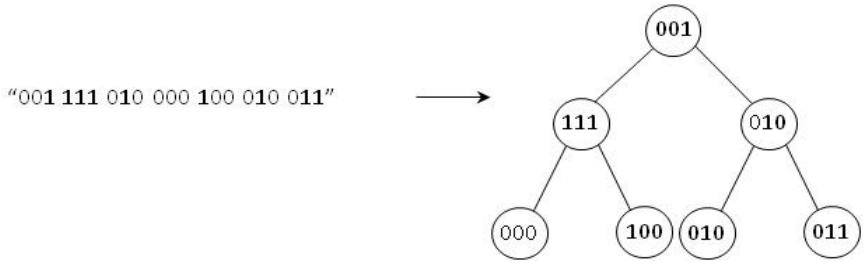

Fig. 1. An illustration of how trees are represented genes

In general, a bit string may not correspond to a fully populated tree and, so such maximal trees need interpretation. Even if we begin with fully constituted trees in the initial population, such trees may arise because of the evolution process described in

the next section. The interpretation used is to assume that the highest parent node that does not correspond to a test is assumed to be a leaf node (i.e. a classification denoting an outcome).

2.2 Evolving the population

Having developed a representation of trees, the next question is: how can the populations evolve? This is done by using the standard crossover and mutation operations. The crossover process takes two of the fittest genes of a population, and combines the first part of one, and the second part of the other to create a new gene. Figure 3 illustrates the crossover operator on two genes representing decision trees. The position of the split is at any random point in the string, but must be on a boundary between nodes. The genes that are chosen for the crossover operation include a random mix of the genes, with greater weighting given to the fittest genes.

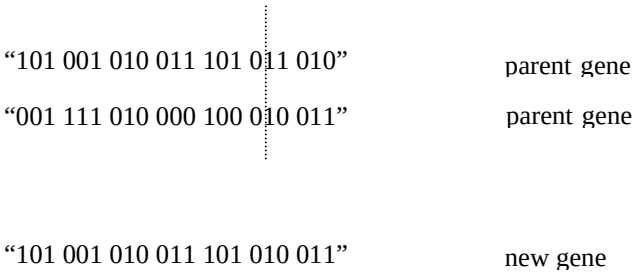

Fig. 2. The crossover operation on representations of trees

The mutation operation changes up to 2% of the nodes. Thus if there are fifty nodes at most in a string, up to one of them will be changed into a different node. This helps prevent the tree from becoming stuck if the fittest genes become identical, and to evolve further to test other slightly different trees.

2.3 Fitness Function

The fitness function has to assess the overall cost of the tree. First the tree needs to be trained to set the classification at each leaf node. That is, a proportion of the data provided is processed through the tree, marking at each node that it has been accessed, and at the classification node the class that row of data belongs to, is appropriately marked. The class chosen at a leaf node is then selected as the one that minimizes the cost of misclassification based on the user provided costs of misclassification.

Once a tree is trained, it is possible to traverse the tree with each example and compute its cost as the sum of the tests used and the cost of any misclassification. The average cost over the examples in the training set is then the fitness function.

3 Empirical comparison with ICET

This section presents an empirical evaluation of ECCO. The primary idea being explored is that utilizing a direct encoding of decision trees for a GA instead of utilizing an intermediate decision tree generator will result in better performance. The ECCO algorithm was implemented by utilizing the GENSIS [20] genetic algorithm and a version of ICET implemented as part of previous work [6] was utilized.

The experiments were designed to test the performance of ECCO under a variety of conditions, and to make comparisons to the performance of ICET. To understand how the algorithms are affected, experiments were first conducted where the cost of misclassification of one class over another were set to: 10, 50, 100, 500, 1000, 5000, 10000. All the experiments were performed with ten iterations using randomly selected training sets consisting of two-thirds of the data and the remaining thirds used as testing data to compute the averages. The depth of trees for ECCO was also varied from 5 to 11.

The maximum depth was chosen as the deepest depth that would give a binary string that was smaller than 50,000 bits in length, with a minimum depth of 5. This bit length was chosen for performance reasons; the larger the string the longer the time needed for the Genetic Algorithm's operations, but a smaller string leads to a more restrictive maximum depth. This value as an upper limit provided a satisfactory balance.

The experiments were performed on the widely used liver, heart, hepatitis, and diabetes benchmark data sets available from the UCI machine learning repository [21]. A range of experiments were carried out, including when misclassifications costs alone are considered, and when both test and misclassification costs are considered. The results had a common pattern for all the data sets and occurs irrespective of whether or not test costs were included. This common pattern is illustrated by Figure 4, which shows the performance of ICET and ECCO (with a depth of 5, with pruning and including both test costs) on the Hepatitis data set. As this illustrates, as the cost of misclassification increases, ECCO is more sensitive and performs better than ICET. The scale of improvement is similar on the other data sets.

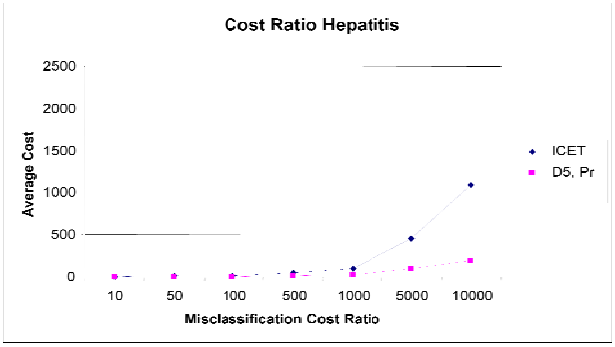

Fig. 3. Average cost as misclassification costs increase for hepatitis data

Given these results, the obvious question to explore is the behaviour of the two approaches when the costs of misclassification are smaller. Figure 5, below, presents the results when the misclassifications cost ratios are varied from 1.5 to 5 and the test costs available in the benchmark data sets are used. The Dn indicates that a tree of depth n is used and results with pruning are indicated with a Pr while results without pruning are indicated with a NPr. These results also show: (a) that ECCO converges to minima more quickly than ICET, (b) that ICET is not very sensitive to increases in misclassification costs, (c) in this range, there isn't much variation in the results whether or not ECCO uses pruning (indeed the differences are not visible in Figure 5 where the lines combine).

4 Conclusion

This paper has introduced a new cost-sensitive decision tree learner, ECCO, that is based on the use genetic algorithms to evolve trees. The algorithm utilizes a more direct coding of trees as genes than the widely cited ICET algorithm that uses biases as genes and utilizes C4.5 and EG2 to generate the trees from the biases. Empirical trials were conducted on four benchmark data sets and the results show that ECCO is more cost-sensitive than ICET to increases in the ratio of costs of misclassification and ECCO produces trees that enable more cost-effective decision making than ICET.

As future work, the performance of ECCO needs to be evaluated on a wider range of benchmark data and against a wider range of cost-sensitive algorithms such as those mentioned in [6,18,19].

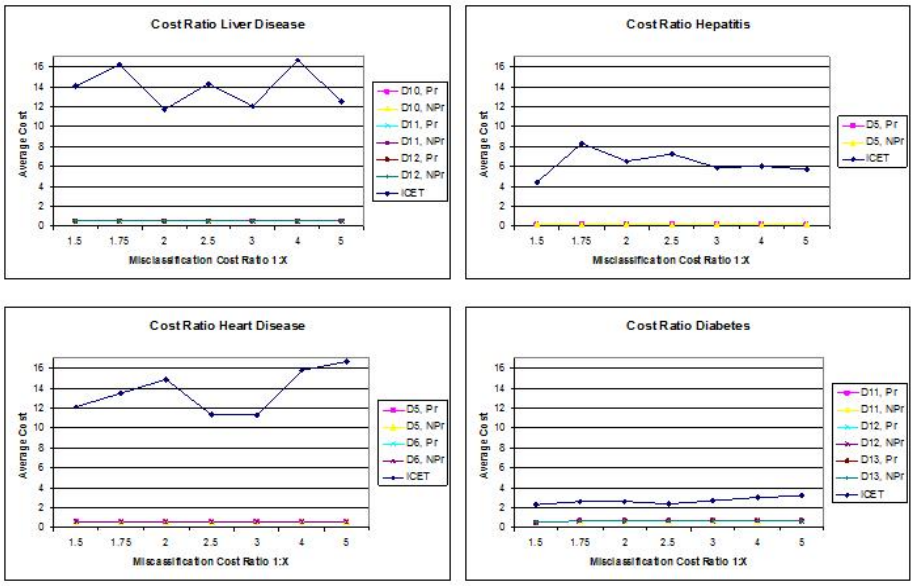

Fig. 4. Results with misclassification costs in the range 1.5 to 5

References

1. QUINLAN, J. R. 1986. Induction of Decision Trees. Machine Learning 1, pp81-106.
2. QUINLAN, J. R. 1993. C4.5: Programs for Machine Learning. Morgan Kaufman, San Mateo, CA.
3. WITTEN, I.H. and FRANK, E. 2005. Data Mining: Practical Machine Learning Tools and Techniques, 2nd Edition, Morgan Kaufmann.
4. BREIMAN, L., FRIEDMAN J. H., OLSEN R. A., STONE C. J. 1984. Classification and Regression Trees. Chapman and Hall/CRC, London
5. 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.
6. VADERA, S. 2010, CSNL A cost-sensitive non-linear decision tree algorithm, ACM Transactions on Knowledge Discovery from Data, Vol 4, No 2, pp1-25.
7. TURNEY, P.D. 2000. Types of cost in inductive concept learning. In Proc. of the Workshop on Cost-Sensitive Learning, 7th Int. Conf. on Machine Learning. 15–21.
8. 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.
9. 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.
10. NÚNEZ, M. 1991. The Use of Background Knowledge in Decision Tree Induction. Machine Learning 6, Kluwer Academic Publishers, Boston, 231-250.
11. 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.
12. LIU, X. 2007. A New Cost-Sensitive Decision Tree with Missing Values. Asian Journal of Information Technology, 6(11), 1083–1090.
13. 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.
14. 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.
15. MORET, S., LANGFORD, W. MARGINEANTU, D. 2006. Learning to predict channel stability using biogeomorphic features. Ecological Modelling, 191(1), 47-57.
16. 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.
17. 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.
18. LOMAX, S., VADERA, S. 2013. A Survey of Cost-Sensitive Decision Tree Induction Algorithms. To appear in ACM Computing Surveys, Vol 45, Issue 2.
19. 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.

20. GREFENSTETTE, J. 1986. Optimization of control parameters for genetic algorithms. IEEE Transactions on Systems, Man, and Cybernetics 16, 122–128.
21. BLAKE, C. And MERZ, C. 1998. UCI Repository of Machine Learning Databases. Irvine, CA:University of California, Department of Information and Computer Science, USA, available at //www.ics.uci.edu/ mlearn/ MLRepository.html.