Anytime Induction of Cost-sensitive Trees

Saher Esmeir
Computer Science Department
Technion—Israel Institute of Technology
Haifa 32000, Israel
esaher@cs.technion.ac.il

Shaul Markovitch
Computer Science Department
Technion—Israel Institute of Technology
Haifa 32000, Israel
shaulm@cs.technion.ac.il

Abstract

Machine learning techniques are increasingly being used to produce a wide-range of classifiers for complex real-world applications that involve nonuniform testing costs and misclassification costs. As the complexity of these applications grows, the management of resources during the learning and classification processes becomes a challenging task. In this work we introduce ACT (Anytime Cost-sensitive Trees), a novel framework for operating in such environments. ACT is an anytime algorithm that allows trading computation time for lower classification costs. It builds a tree top-down and exploits additional time resources to obtain better estimations for the utility of the different candidate splits. Using sampling techniques ACT approximates for each candidate split the cost of the subtree under it and favors the one with a minimal cost. Due to its stochastic nature ACT is expected to be able to escape local minima, into which greedy methods may be trapped. Experiments with a variety of datasets were conducted to compare the performance of ACT to that of the state of the art cost-sensitive tree learners. The results show that for most domains ACT produces trees of significantly lower costs. ACT is also shown to exhibit good anytime behavior with diminishing returns.

1 Introduction

Suppose that a medical center has decided to use machine learning techniques to induce a diagnostic tool from records of previous patients. The center aims to obtain a comprehensible model, with low expected *test costs* (the costs of testing attribute values) and high expected accuracy. Moreover, in many cases there are costs associated with the predictive errors. In such a scenario, the task of the inducer is to produce a model with low expected test costs and low expected *misclassification costs*.

A good candidate for achieving the goals of comprehensibility and reduced costs is a decision tree model. Decision trees are easily interpretable because they mimic the way doctors think [13][chap. 9]. In the context of cost-sensitive classification, decision trees are the natural form of representation: they ask only for the values of the features along a single path from the root to a leaf. Indeed, cost-sensitive trees have been the subject of many research efforts. Several works proposed learners that consider different misclassification costs [7, 18, 6, 9, 10, 14, 1]. These methods, however, do not consider test costs. Other authors designed tree learners that take into account test costs, such as IDX [16], CSID3 [22], and EG2 [17]. These methods, however, do not consider misclassification costs. The medical center scenario exemplifies the need for considering both types of cost together: doctors do not perform a test before considering both its cost and its importance to the diagnosis.

Minimal Cost trees, a method that attempts to minimize both types of costs simultaneously has been proposed in [21]. A tree is built top-down. The immediate reduction in total cost each split results in is estimated, and a split with the maximal reduction is selected. Although efficient, the Minimal Cost approach can be trapped into a local minimum and produce trees that are not globally optimal.

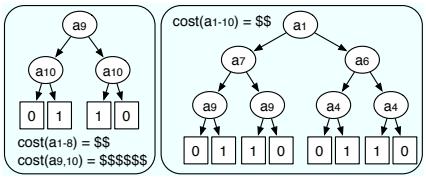

Figure 1: A difficulty for greedy learners (left). Importance of context-based evaluation (right).

For example, consider a problem with 10 attributes a_{1-10}, of which only a_9 and a_{10} are relevant. The cost of a_9 and a_{10}, however, is significantly higher than the others but lower than the cost of misclassification. This may hide their usefulness, and mislead the learner to fit a large expensive tree. The problem is intensified if a_9 and a_{10} were interdependent with a low immediate information gain (e.g., $a_9 \oplus a_{10}$), as illustrated in Figure 1 (left). In such a case, even if the costs were uniform, local measures would fail in recognizing the relevance of a_9 and a_{10} and other attributes might be preferred. The Minimal Cost method is appealing when resources are very limited. However, it requires a fixed runtime and cannot exploit additional resources. In many real-life applications, we are willing to wait longer if a better tree can be induced. For example, due to the importance of the model, the medical center is ready to allocate 1 week to learn it. Algorithms that can exploit more time to produce solutions of better quality are called anytime algorithms [5].

One way to exploit additional time when searching for a tree of lower costs is to widen the search space. In [2] the cost-sensitive learning problem is formulated as a Markov Decision Process (MDP) and a systematic search is used to solve the MDP. Although the algorithm searches for an optimal strategy, the time and memory limits prevent it from always finding optimal solutions.

The ICET algorithm [24] was a pioneer in searching non-greedily for a tree that minimizes both costs together. ICET uses genetic search to produce a new set of costs that reflects both the original costs and the contribution each attribute can make to reduce misclassification costs. Then it builds a tree using the greedy EG2 algorithm but with the evolved costs instead of the original ones. ICET was shown to produce trees of lower *total cost*. It can use additional time resources to produce more generations and hence to widen its search in the space of costs. Nevertheless, it is limited in the way it can exploit extra time. Firstly, it builds the final tree using EG2. EG2 prefers attributes with high information gain (and low test cost). Therefore, when the concept to learn hides interdependency between attributes, the greedy measure may underestimate the usefulness of highly relevant attributes, resulting in more expensive trees. Secondly, even if ICET may overcome the above problem by reweighting the attributes, it searches the space of parameters globally, regardless of the context. This imposes a problem if an attribute is important in one subtree but useless in another. To illustrate the above consider the concept in Figure 1 (right). There are 10 attributes of similar costs. Depending on the value of a_1, the target concept is $a_7 \oplus a_9$ or $a_4 \oplus a_6$. Due to interdependencies, all attributes will have a low gain. Because ICET assigns costs globally, they will have similar costs as well. Therefore, ICET will not be able to recognize which attribute is relevant in what context.

Recently, we have introduced LSID3, a cost-insensitive algorithm, which can induce more accurate trees when given more time [11]. The algorithm uses stochastic sampling techniques to evaluate candidate splits. It is not designed, however, to minimize test and misclassification costs. In this work we build on LSID3 and propose *ACT*, an Anytime Cost-sensitive Tree learner that can exploit additional time to produce trees of lower costs. Applying the sampling mechanism to the cost-sensitive setup, however, is not trivial and imposes several challenges which we address in Section 2. Extensive set of experiments that compares ACT to EG2 and to ICET is reported in Section 3. The results show that ACT is significantly better for the majority of problems. In addition ACT is shown to exhibit good anytime behavior with diminishing returns. The major contributions of this paper are: (1) a non-greedy algorithm for learning trees of lower costs that allows handling complex cost structures, (2) an anytime framework that allows learning time to be traded for reduced classification costs, and (3) a parameterized method for automatic assigning of costs for existing datasets.

Note that costs may also be involved during example acquisition [12, 15]. In this work, however, we assume that the full training examples are in hand. Moreover, we assume that during the test phase, all tests in the relevant path will be taken. Several test strategies that determine which values to query for and at what order have been recently studied [21]. These strategies are orthogonal to our work because they assume a given tree.

2 The ACT Algorithm

Offline concept learning consists of two stages: learning from labelled examples; and using the induced model to classify unlabelled instances. These two stages involve different types of cost [23]. Our primary goal in this work is to trade the learning time for reduced test and misclassification costs. To make the problem well defined, we need to specify how to: (1) represent misclassification costs, (2) calculate test costs, and (3) combine both types of cost.

To answer these questions, we adopt the model described by Turney [24]. In a problem with $|C|$ different classes, a classification cost matrix M is a $|C| \times |C|$ matrix whose $M_{i,j}$ entry defines the penalty of assigning the class c_i to an instance that actually belongs to the class c_j. To calculate the test costs of a particular case, we sum the cost of the tests along the path from the root to the appropriate leaf. For tests that appear several times we charge only for the first occurrence. The model handles two special test types, namely *grouped* and *delayed*. Grouped tests share a common cost that is charged only once per group. Each test also has an extra cost charged when the test is actually made. For example, consider a tree path with tests like cholesterol level and glucose level. For both values to be measured, a blood test is needed. Clearly, once blood samples are taken to measure the cholesterol level, the cost for measuring the glucose level is lower. Delayed tests are tests whose outcome cannot be obtained immediately, e.g., lab test results. Such tests force us to wait until the outcome is available. Alternatively, we can take into account all possible outcomes and follow several paths in the tree simultaneously (and pay for their costs). Once the result of the delayed test is available, the prediction is in hand. Note that we might be charged for tests that we would not perform if the outcome of the delayed tests were available. In this work we do not handle delayed costs but we do explain how to adapt our framework to scenarios that involve them.

Having measured the test costs and misclassification costs, an important question is how to combine them. Following [24] we assume that both types of cost are given in the same scale. Alternatively, Qin et. al. [19] presented a method to handle the two kinds of cost scales by setting a maximal budget for one kind and minimizing the other.

ACT, our proposed anytime framework for induction of cost-sensitive trees, builds on the recently introduced LSID3 algorithm [11]. LSID3 adopts the general top-down induction of decision trees scheme (TDIDT): it starts from the entire set of training examples, partitions it into subsets by testing the value of an attribute, and then recursively builds subtrees. Unlike greedy inducers, LSID3 invests more time resources for making better split decisions. For every candidate split, LSID3 attempts to estimate the size of the resulting subtree were the split to take place and following Occam's razor [4] it favors the one with the smallest expected size. The estimation is based on a biased sample of the space of trees rooted at the evaluated attribute. The sample is obtained using a stochastic version of ID3, called SID3 [11]. In SID3, rather than choosing an attribute that maximizes the information gain ΔI (as in ID3), the splitting attribute is chosen semi-randomly. The likelihood that an attribute will be chosen is proportional to its information gain. LSID3 is a contract algorithm parameterized by r, the sample size. When r is larger, the resulting estimations are expected to be more accurate, therefore improving the final tree. Let $m = |E|$ be the number of examples and $n = |A|$ be the number of attributes. The runtime complexity of LSID3 is $O(rmn^3)$ [11]. LSID3 was shown to exhibit a good anytime behavior with diminishing returns. When applied to hard concepts, it produced significantly better trees than ID3 and C4.5. ACT takes the same sampling approach as in LSID3. However, three major components of LSID3 need to be replaced for the cost-sensitive setup: (1) sampling the space of trees, (2) evaluating a tree, and (3) pruning.

Obtaining the Sample. LISD3 uses SID3 to bias the samples towards small trees. In ACT, however, we would like to bias our sample towards low cost trees. For this purpose, we designed a stochastic version of the EG2 algorithm, that attempts to build low cost trees greedily. In EG2, a tree is built top-down, and the attribute that maximizes ICF (Information Cost Function) is chosen for splitting a node, where, $\text{ICF}(a) = \left(2^{\Delta I(a)} - 1\right) / \left((\text{cost}(a) + 1)^w\right)$.

In Stochastic EG2 (SEG2), we choose splitting attributes semi-randomly, proportionally to their ICF. Due to the stochastic nature of SEG2 we expect to be able to escape local minima for at least some of the trees in the sample. To obtain a sample of size r, ACT uses EG2 once and SEG2 $r - 1$ times. Unlike ICET, we give EG2 and SEG2 a direct access to context-based costs, i.e., if an attribute has already been tested its cost would be zero and if another attribute that belongs to the same group has been tested, a group discount is applied. The parameter w controls the bias towards lower cost

attributes. While ICET tunes this parameter using genetic search, we set w inverse proportionally to the misclassification cost: a high misclassification cost results in a smaller w, reducing the effect of attribute costs. One direction for future work would be to tune w a priori.

Evaluating a Subtree. As a cost insensitive learner, the main goal of LSID3 is to maximize the expected accuracy of the learned tree. Following Occam's razor, it uses the tree size as a preference bias and favors splits that are expected to reduce the final tree size. In a cost-sensitive setup, our goal is to minimize the expected cost of classification. Following the same lookahead strategy as LSID3, we sample the space of trees under each candidate split. However, instead of choosing an attribute that minimizes the size, we would like to choose one that minimizes costs. Therefore, given a tree, we need to come up with a procedure that estimates the expected costs when classifying a future case. This cost consists of two components: the test cost and misclassification cost.

Assuming that the distribution of future cases would be similar to that of the learning examples, we can estimate the test costs using the training data. Given a tree, we calculate the average test cost of the training examples and use it to approximate the test cost of new cases. For a tree T and a set of training examples E, we denote the average cost of traversing T for an example from E (average testing cost) by tst-cost(T, E). Note that group discounts and delayed cost penalties do not need a special care because they will be incorporated when calculating the average test costs.

Estimating the cost of errors is not obvious. One can no longer use the tree size as a heuristic for predictive errors. Occam's razor allows to compare two consistent trees but does not provide a mean to estimate accuracy. Moreover, tree size is measured in a different currency than accuracy and hence cannot be easily incorporated in the cost function. Instead, we propose using a different estimator: the expected error [20]. For a leaf with m training examples, of which e are misclassified the expected error is defined as the upper limit on the probability for error, i.e., $EE(m, e, cf) = U_{cf}(e, m)$ where cf is the confidence level and U is the confidence interval for binomial distribution. The expected error of a tree is the sum of the expected errors in its leafs. Originally, the expected error was used by C4.5 to predict whether a subtree performs better than a leaf. Although it lacks theoretical basis, it was shown experimentally to be a good heuristic. In ACT we use the expected error to approximate the misclassification cost. Assume a problem with $|C|$ classes and a misclassification cost matrix M. Let c be the class label in a leaf l. Let m be the total number of examples in l and m_i be the number of examples in l that belong to class i. The expected misclassification cost in l is (the right most expression assumes uniform misclassification cost $M_{i,j} = mc$)

$$\text{mc-cost}(l) = EE(m, m - m_c, cf) \cdot \frac{1}{|C| - 1} \sum_{i \neq c} M_{c,i} = EE(m, m - m_c, cf) \cdot mc$$

The expected error of a tree is the sum of the expected errors in its leafs. In our experiments we use $cf = 0.25$, as in C4.5. In the future, we intend to tune cf if the allocated time allows. Alternatively, we also plan to estimate the error using a set-aside validation set, when the training set size allows. To conclude, let E be the set of examples used to learn a tree T, and let m be the size of E. Let L be the set of leafs in T. The expected total cost of T when classifying an instance is:

$$\text{tst-cost}(T, E) + \frac{1}{m} \cdot \sum_{l \in L} \text{mc-cost}\,(l).$$

Having decided about the sampler and the tree utility function we are ready to formalize the tree growing phase in ACT. A tree is built top-down. The procedure for selecting splitting test at each node is listed in Figure 2 (left), and exemplified in Figure 2 (right). The selection procedure, as formalized is Figure 2 (left) needs to be slightly modified when an attribute is numeric: instead of iterating over the values the attribute can take, we examine r cutting points, each is evaluated with a single invocation of EG2. This guarantees that numeric and nominal attributes get the same resources. The r points are chosen dynamically, according to their information gain.

Costs-sensitive Pruning. Pruning plays an important role in decision tree induction. In cost-insensitive environments, the main goal of pruning is to simplify the tree in order to avoid overfitting. A subtree is pruned if the resulting tree is expected to yield a lower error. When test costs are taken into account, pruning has another important role: reducing costs. It is worthwhile to keep a subtree only if its expected reduction to the misclassification cost is larger that the cost of its tests. If the misclassification cost was zero, it makes no sense to keep any split in the tree. If, on the other hand,

Procedure ACT-CHOOSE-ATTRIBUTE(E, A, r)
  If $r = 0$ **Return** EG2-CHOOSE-ATTRIBUTE(E, A)
  Foreach $a \in A$
    Foreach $v_i \in \text{domain}(a)$
      $E_i \leftarrow \{e \in E \mid a(e) = v_i\}$
      $T \leftarrow \text{EG2}(a, E_i, A - \{a\})$
      $min_i \leftarrow \text{COST}(T, E_i)$
      Repeat $r - 1$ **times**
        $T \leftarrow \text{SEG2}(a, E_i, A - \{a\})$
        $min_i \leftarrow \min(min_i, \text{COST}(T, E_i))$
    $total_a \leftarrow \text{COST}(a) + \sum_{i=1}^{|\text{domain}(a)|} min_i$
  Return a for which $total_a$ is minimal

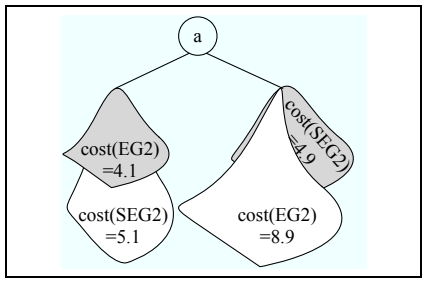

Figure 2: Attribute selection (left) and evaluation (right) in ACT (left). Assume that the cost of a in the current context is 1. The estimated cost of a subtree rooted at a is therefore $1 + min(4.1, 5.1) + min(8.9, 4.9) = 9$.

the misclassification cost was very large, we would expect similar behavior to the cost-insensitive setup. To handle this challenge, we propose a novel approach for cost-sensitive pruning. Similarly to error-based pruning [20], we scan the tree bottom-up. For each subtree, we compare its expected total cost to that of a leaf. Formally, assume that e examples in E do not belong to the default class.[1] We prune a subtree T into a leaf if:

$$\frac{1}{m} \cdot \text{mc-cost}(l) \leq \text{tst-cost}(T, E) + \frac{1}{m} \cdot \sum_{l \in L} \text{mc-cost}(l).$$

3 Empirical Evaluation

A variety of experiments were conducted to test the performance and behavior of ACT. First we describe and motivate our experimental methodology. We then present and discuss our results.

3.1 Methodology

We start our experimental evaluation by comparing ACT, given a fixed resource allocation, with EG2 and ICET. EG2 was selected as a representative for greedy learners. We also tested the performance of CSID3 and IDX but found the results very similar to EG2, confirming the report in [24]. Our second set of experiments compares the anytime behavior of ACT to that of ICET. Because the code of EG2 and ICET is not publicly available we have reimplemented them. To verify the reimplementation results, we compared them with those reported in literature. We followed the same experimental setup and used the same 5 datasets. The results are indeed similar with the basic version of ICET achieving an average cost of 49.9 in our reimplementation vs. 49 in Turney's paper [24]. One possible reason for the slight difference may be the randomization involved in the genetic search as well as in data partitioning into training, validating, and testing sets.

Datasets. Typically, machine learning researchers use datasets from the UCI repository [3]. Only five UCI datasets, however, have assigned test costs [24]. To gain a wider perspective, we developed an automatic method that assigns costs to existing datasets randomly. The method is parameterized with: (1) cr the cost range, (2) g the number of desired groups as a percentage of the number of attributes, and (3) sc the group shared cost as a percentage of the maximal marginal cost in the group. Using this method we assigned costs to 25 datasets: 21 arbitrarily chosen UCI datasets[2] and 4 datasets that represent hard concept and have been used in previous research. The online appendix [3] gives detailed descriptions of these datasets. Two versions of each dataset have been created, both with cost range of 1-100. In the first g and sc were set to 20% and in the second they were set to 80%. These parameters were chosen arbitrarily, in attempt to cover different types of costs. In total we have 55 datasets: 5 with costs assigned as in [24] and 50 with random costs. Cost-insensitive learning algorithms focus on accuracy and therefore are expected to perform well

[1] The default class is the one that minimizes the misclassification cost in the node.

[2] The chosen UCI datasets vary in their size, type of attributes and dimension.

[3] http://www.cs.technion.ac.il/∼esaher/publications/nips07

Table 1: Average cost of classification as a percentage of the standard cost of classification. The table also lists for each of ACT and ICET the number of significant wins they had using t-test. The last row shows the winner, if any, as implied by a Wilcoxon test over all datasets with $\alpha = 5\%$.

	$mc = 10$			$mc = 100$			$mc = 1000$			$mc = 10000$		
	EG2	ICET	ACT	EG2	ICET	ACT	EG2	ICET	ACT	EG2	ICET	ACT
AVERAGE	22.37	10.23	2.21	25.93	17.15	11.86	38.69	35.28	34.38	54.22	47.47	41.62
BETTER		0	34		0	25		3	11		10	12
WILCOXON			$\surd$			$\surd$						$\surd$

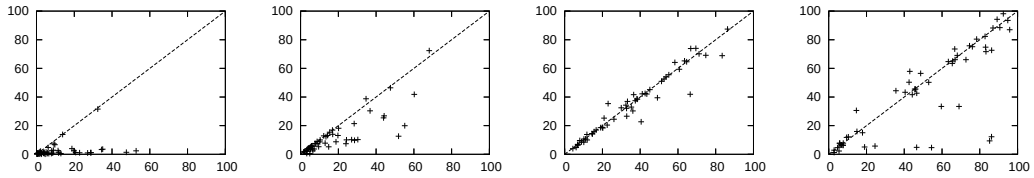

Figure 3: Illustration of the differences in performance between ACT and ICET for misclassification costs (from left to right: 10, 100, 1000, and 10000). Each point represents a dataset. The x-axis represents the cost of ICET while the y-axis represents that of ACT. The dashed line indicates equality. Points are below it if ACT performs better and above it if ICET is better.

when testing costs are negligible relative to misclassification costs. On the other hand, when testing costs are significant, ignoring them would result in expensive classifiers. Therefore, to evaluate a cost-sensitive learner a wide spectrum of misclassification costs is needed. For each problem out of the 55, we created 4 instances, with uniform misclassification costs $mc = 10, 100, 1000, 10000$.

Normalized Cost. As pointed out by Turney [24], using the average cost is problematic because: (1) the differences in costs among the algorithms become small as misclassification cost increases, (2) it is difficult to combine the results for the multiple datasets, and (3) it is difficult to combine average costs for different misclassification costs. To overcome these problems, Turney suggests to normalize the average cost of classification by dividing it by the *standard cost*, defined as $(TC + min_i\,(1 - f_i) \cdot max_{i,j}\,(M_{i,j}))$, The standard cost is an approximation for the maximal cost in a given problem. It consists of two components: (1) TC, the cost if we take all tests, and (2) the misclassification cost if the classifier achieves only the base-line accuracy. f_i denotes the frequency of class i in the data and hence $(1 - f_i)$ would be the error if the response would always be class i.

Statistical Significance. For each problem, one 10 fold cross-validation experiment has been conducted. The same partition to train-test sets was used for all compared algorithms. To test the statistical significance of the differences between ACT and ICET we used two tests. The first is t-test with a $\alpha = 5\%$ confidence: for each method we counted how many times it was a significant winner. The second is Wilcoxon test [8], which compares classifiers over multiple datasets and states whether one method is significantly better than the other ($\alpha = 5\%$).

3.2 Fixed-time Comparison

For each of the 55×4 problem instances, we run the seeded version of ICET with its default parameters (20 generations),[4] EG2, and ACT with $r = 5$. We choose $r = 5$ so the average runtime of ACT would be shorter than ICET for all problems. EG2 and ICET use the same post-pruning mechanism as in C4.5. In EG2 the default confidence factor is used (0.25) while in ICET this value is tuned using the genetic search.

Table 1 lists the average results, Figure 3 illustrates the differences between ICET and ACT, and Figure 4 (left) plots the average cost for the different values of mc. The full results are available in the online appendix. Similarly to the results reported in [24] ICET is clearly better than EG2, because the latter does not consider misclassification costs. When mc is set to 10 and to 100 ACT significantly outperforms ICET for most datasets. In these cases ACT was able to produce very small trees, sometimes consist of one node, neglecting the accuracy of the learned model. For mc set to 1000 and 10000 there are fewer significant wins, yet it is clear that ACT is dominating: the

[4]Seeded ICET includes the true costs in the initial population and was reported to perform better [24].

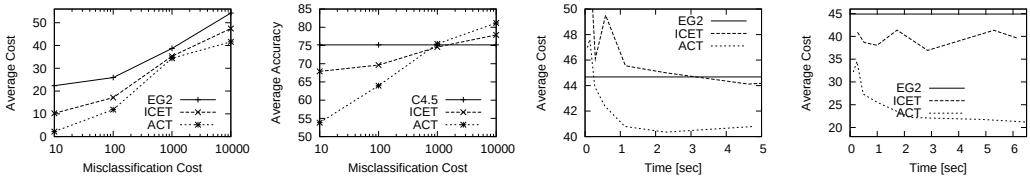

Figure 4: Average cost (left most) and accuracy (mid-left) as a function of misclassification cost. Average cost as a function of time for Breast-cancer-20 (mid-right) and Multi-XOR-80 (right most).

number of ACT wins is higher and the average results indicate that ACT trees are cheaper. The Wilcoxon test, states that for $mc = 10, 100, 10000$, ACT is significantly better than ICET, and that for $mc = 1000$ no significant winner was found.

When misclassification costs are low, an optimal algorithm would produce a very shallow tree. When misclassification costs are dominant, an optimal algorithm would produce a highly accurate tree. Some concepts, however, are not easily learnable and even cost-insensitive algorithms fail to achieve perfect accuracy on them. Hence, with the increase in the importance of accuracy the normalized cost increases: the predictive errors affect the cost more dramatically. To learn more about the effect of accuracy, we compared the accuracy of ACT to that of C4.5 and ICET mc values. Figure 4 (mid-left) shows the results. An important property of both ICET and ACT is their ability to compromise on accuracy when needed. ACT's flexibility, however, is more noteworthy: from the least accurate method it becomes the most accurate one. Interestingly, when accuracy is extremely important both ICET and ACT achieves even better accuracy than C4.5. The reason is their non-greedy nature. ICET performs an implicit lookahead by reweighting attributes according to their importance. ACT performs lookahead by sampling the space of subtrees under every split. Among the two, the results indicates that ACT's lookahead is more efficient in terms of accuracy. We also compared ACT to LSID3. As expected, ACT was significantly better for $mc \leq 1000$. For $mc = 10000$ their performance was similar. In addition, we compared the studied methods on nonuniform misclassification costs and found ACT's advantage to be consistent.

3.3 Anytime Comparison

Both ICET and ACT are anytime algorithms that improve their performance with time. ICET is expected to exploit extra time by producing more generations and hence better tuning the parameters for the final invocation of EG2. ACT can use additional time to acquire larger samples and hence achieve better cost estimations. A typical anytime algorithm would produce improved results with the increase in resources. The improvements diminish with time, reaching a stable performance.

To examine the anytime behavior of ICET and ACT, we run each of them on 2 problems, namely Breast-cancer-20 and Multi-XOR-80, with exponentially increasing time allocation. ICET was run with $2, 4, 8 \ldots$ generations and ACT with a sample size of $1, 2, 4, \ldots$. Figure 4 plots the results. The results show a good anytime behavior of both ICET and ACT. For both algorithms, it is worthwhile to allocate more time. ACT dominates ICET for both domains and is able to produce trees of lower costs in shorter time. The Multi-XOR dataset is an example for a concept with attributes being important only in one sub-concept. As we expected, ACT outperforms ICET significantly because the latter cannot assign context-based costs. Allowing ICET to produce more and more generations (up to 128) does not result in trees comparable to those obtained by ACT.

4 Conclusions

Machine learning techniques are increasingly being used to produce a wide-range of classifiers for real-world applications that involve nonuniform testing costs and misclassification costs. As the complexity of these applications grows, the management of resources during the learning and classification processes becomes a challenging task. In this work we introduced a novel framework for operating in such environments. Our framework has 4 major advantages: (1) it uses a non-greedy approach to build a decision tree and therefore is able to overcome local minima problems, (2) it evaluates entire trees and therefore can be adjusted to any cost scheme that is defined over trees. (3) it exhibits good anytime behavior and produces significantly better trees when more time is available, and (4) it can be easily parallelized and hence can benefit from distributed computer power.

To evaluate ACT we have designed an extensive set of experiments with a wide range of costs. The experimental results show that ACT is superior over ICET and EG2. Significance tests found the differences to be statistically strong. ACT also exhibited good anytime behavior: with the increase in time allocation, there was a decrease in the cost of the learned models. ACT is a contract anytime algorithm that requires its sample size to be pre-determined. In the future we intend to convert ACT into an interruptible anytime algorithm, by adopting the IIDT general framework [11]. In addition, we plan to apply monitoring techniques for optimal scheduling of ACT and to examine other strategies for evaluating subtrees.

References

[1] N. Abe, B. Zadrozny, and J. Langford. An iterative method for multi-class cost-sensitive learning. In *KDD*, 2004.

[2] V. Bayer-Zubek and Dietterich. Integrating learning from examples into the search for diagnostic policies. *Artificial Intelligence*, 24:263–303, 2005.

[3] C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998.

[4] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Occam's Razor. *Information Processing Letters*, 24(6):377–380, 1987.

[5] M. Boddy and T. L. Dean. Deliberation scheduling for problem solving in time constrained environments. *Artificial Intelligence*, 67(2):245–285, 1994.

[6] J. Bradford, C. Kunz, R. Kohavi, C. Brunk, and C. Brodley. Pruning decision trees with misclassification costs. In *ECML*, 1998.

[7] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[8] J. Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[9] P. Domingos. Metacost: A general method for making classifiers cost-sensitive. In *KDD*, 1999.

[10] C. Elkan. The foundations of cost-sensitive learning. In *IJCAI*, 2001.

[11] S. Esmeir and S. Markovitch. Anytime learning of decision trees. *Journal of Machine Learning Research*, 8, 2007.

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

[13] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer-Verlag, 2001.

[14] D. Margineantu. Active cost-sensitive learning. In *IJCAI*, 2005.

[15] P. Melville, M. Saar-Tsechansky, F. Provost, and R. J. Mooney. Active feature acquisition for classifier induction. In *ICDM*, 2004.

[16] S. W. Norton. Generating better decision trees. In *IJCAI*, 1989.

[17] M. Nunez. The use of background knowledge in decision tree induction. *Machine Learning*, 6:231–250, 1991.

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

[19] Z. Qin, S. Zhang, and C. Zhang. Cost-sensitive decision trees with multiple cost scales. *Lecture Notes in Computer Science, AI*, Volume 3339/2004:380–390, 2004.

[20] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[21] S. Sheng, C. X. Ling, A. Ni, and S. Zhang. Cost-sensitive test strategies. In *AAAI*, 2006.

[22] M. Tan and J. C. Schlimmer. Cost-sensitive concept learning of sensor use in approach and recognition. In *Proceedings of the 6th international workshop on Machine Learning*, 1989.

[23] P. Turney. Types of cost in inductive concept learning. In *Workshop on Cost-Sensitive Learning at ICML*, 2000.

[24] 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.