Anytime Algorithms for Learning Resource-bounded Classifiers

Keywords: anytime algorithms, resource-bounded reasoning, cost-sensitive learning, decision trees

Saher Esmeir ESAHER@CS.TECHNION.AC.IL
DRW Trading Group

Shaul Markovitch SHAULM@CS.TECHNION.AC.IL
Department of Computer Science, Technion

Abstract

The classification of new cases using a predictive model incurs two types of costs—testing and misclassification. Recent research efforts have resulted in several algorithms that attempt to minimize both costs simultaneously. In many real life scenarios, however, we cannot afford to conduct all the tests required by the predictive model. Due to their structure, decision trees are considered attractive for cost-bounded classification as they measure only the tests along a single path. In this work we present an anytime framework for producing tree-based classifiers that can make accurate decisions within a strict bound on testing costs. These bounds can be known to the learner, known to the classifier but not to the learner, or not predetermined. Extensive experiments with a variety of datasets show that our proposed framework produces trees of lower misclassification costs along a wide range of testing cost bounds.

1. Introduction

Assume that a hardware manufacturer has decided to use a machine-learning based tool for assuring the quality of produced chips. In realtime, each chip in the pipeline is scanned and several features can be extracted from the image. The features vary in their computation time. The manufacturer trains the component using thousands of chips whose validity is known. Because the training is made offline, the manufacturer can provide the values of all possible features, regardless of their computation time. When used in realtime, however, the model must make a decision within 2 seconds. Therefore, for each chip, the classifier may use features whose total computation time is at most 2 seconds. A similar situation may occur in medical diagnoses. In many cases, for example due to limits on insurance coverage, there is a fixed, possibly different, budget for diagnosing each patient. A third scenario is when we do not know the bound on resources in advance. Consider, for example, a real-time classifier for detecting network attack traffic. Based on the current load, the available resources to classify a packet vary, and the model must be prepared to return a prediction upon the arrival of the next packet.

An algorithm that improves the quality of its output with the increase in allocated resources is called an anytime algorithm (Boddy & Dean, 1994). We use a similar notation and refer to a classifier whose expected misclassification costs decrease with the increase in allocation of testing resources as an *anycost* classifier. One way to produce an anycost classifier is to select a subset of the features whose total cost fits the budget and to build a model using features from this set. Selecting a single subset may be helpful when the learner is aware of the bound on classification cost, but not when this bound is determined after learning the model. Moreover, some features might be irrelevant for some cases. Counting their cost in the quota prevents the learner from using other relevant ones.

An alternative is to use tree-based classifiers. In addition to their comprehensibility and ease of use, decision trees are considered attractive when classification costs need to be controlled because they ask only for the values of the features along a single path from the root to a leaf. Decision trees can be easily converted into anycost classifiers. If the learner stores at each node a default class label, the classification procedure can be terminated at any time. However, solutions that do not consider the test costs during learning may result in prohibitively expensive trees. Consider,

Appearing in *Proceedings of the Budgeted Learning Workshop, in conjunction with ICML-2010*, Haifa, Israel, 2010.

for example, a tree with some very expensive tests at the top levels. If the classification budget is relatively low, traversing the decision path would be impossible and the classifier would have to stop too early and return a prediction. This situation could be avoided if the learner considers testing costs when choosing the splitting tests in the tree.

Recently, several researchers have proposed cost-sensitive decision-tree learners that attempt to minimize the total cost during classification, that is, the sum of testing costs and misclassification costs. Among these classifiers are ICET (Inexpensive Classification with Expensive Tests, Turney 1995), DTMC (Decision Trees with Minimal Cost, Sheng et al. 2006), and ACT (Anytime learning of Cost-sensitive Trees, Esmeir & Markovitch 2007a). While successful in minimizing the total cost and producing cost-effcient trees, cost-sensitive learners are not designed to build trees that operate under strict bounds on classification resources. As a result, the produced classifier may exceed the classification budget or underuse it. What is needed is a learning algorithm for producing decision trees that can operate under strict classification budgets and yield low misclassification costs.

Greiner et al. (2002) were pioneers in studying classifiers that actively decide what tests to administer. They defined an *active classifier* as a classifier that given a partially specified instance, returns either a class label or a strategy that specifies which test should be performed next. Greiner et al. also analyzed the theoretical aspects of learning optimal active classifiers using a variant of the PAC model. They showed that the task of learning optimal cost-sensitive active classifiers is often intractable. However, this task is shown to be achievable when the classifier is allowed to perform at most a constant number of tests, where the limit is provided before learning. For this setup they proposed taking a dynamic programming approach to build decision trees of at most depth d. Their proposed method requires $O(n^d)$ time to find the optimal d-depth tree over n attributes. Although this complexity is polynomial in n, it can be used, in practice, only when n and d are small. Kapoor & Greiner (2005) extended this work to support predetermined limits on test costs rather than number of tests. While these works can be applied when the budget is determined before learning, they are not designed to be applied when the maximal cost is not available to the learner.

To address these challenges we introduce TATA (Tree-classification AT Anycost), a novel framework for resource-bounded learning and classification. TATA is an anytime learner that can utilize additional time to produce anycost classifiers that can exploit additional classification resources. TATA is general and can be adapted to any time-cost scheme. Extensive experimental evaluation with several datasets and under a variety of setups indicates that TATA performs better than existing alternatives and exhibits good anytime and anycost behavior.

2. Cost-sensitive Classification

Offline concept learning consists of several stages, each of which involves several types of cost (Turney, 2000). In the first stage the data is collected. This includes acquiring the values of the different features and labeling the examples. Several works have tackled the questions of cost sensitive example and feature-value acquisition (Melville et al., 2004; Lizotte et al., 2003; Sheng & Ling, 2007; Bilgic & Getoor, 2007).

Once the data has been collected, the next stage is model learning. This stage usually demands resources such as memory and CPU time. Finally, during the classification phase, costs might be associated with the measurements required by the model and with the prediction errors. In many real-life scenarios we are willing to allocate more resources during learning to save costs later on when using the model predictions. Therefore, our primary goal in this work is to use more learning resources in order to offer flexible anycost classifiers that can trade testing costs for reduced misclassification costs.

Following previous works on cost-sensitive induction, we adopt the model described by Turney (1995). In a problem with $|C|$ different classes, a misclassification 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 belongs to the class c_j. Let ρ be the bound on testing costs for each case. When classifying an example e using a tree T, we propagate e down the tree until a leaf is reached or classification resources are exhausted. Let $\Theta(T, e)$ be the set of tests along the classification path, and $cost(\theta)$ be the cost of administering the test θ. The testing cost of e in T is therefore $tcost(T, e) = \sum_{\theta \in \Theta} cost(\theta)$. By definition $tcost(T, e) \leq \rho$. Unlike the model of Turney (1995), we do not require $M_{i,j}$ and $cost(\theta)$ to be provided in the same currency.

In this work we provide a framework that can operate under three different cost scenarios: (1) *Pre-contract*: ρ is known before learning and the inducer aims to build a tree that operates within a budget of ρ and minimizes misclassification costs, (2) *contract*: ρ is not provided to the learner but is known before proceed-

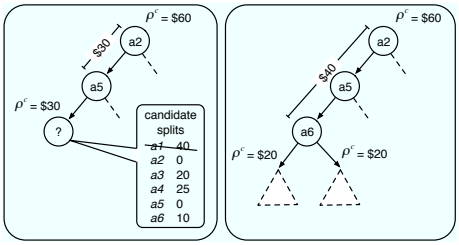

Figure 1. Recursive invocation of TDIDT$

ing with classification, thus the learner must produce a classifier that can operate under any provided ρ, and (3) *interruptible*: ρ is not pre-determined so the classifier needs to utilize resources until interrupted.

By being able to exploit additional learning time, our framework is also an anytime learner. Although the learning process itself can be contract or interruptible, in this work we focus on contract anytime learning and assume pre-allocated learning resources.

3. The TATA Algorithm

TATA, our proposed anytime algorithm for learning anycost trees, adopts the general top-down induction of decision trees scheme (TDIDT). TDIDT algorithms start from the entire set of training examples, partition it into subsets by testing the value of an attribute, and then recursively build subtrees. We first describe a modification for adapting any TDIDT algorithm to the *pre-contract* scenario. We then introduce the *pre-contract* component in the TATA framework. Next, we present a general technique for converting *pre-contract* tree learners into *contract* and *interruptible* ones, and discuss how it can be implemented in TATA.

3.1. Top-down Induction of Anycost Trees

Any decision tree can serve as an anycost classifier by simply storing a default classification at each node and predicting according to the last explored node. Expensive tests, in this case, may block their subtrees and make them useless. In the *pre-contract* setup, ρ, the maximal testing cost, is known in advance and thus the tree learner can avoid exceeding ρ by considering only tests whose costs are lower. When a test θ is chosen to split on, the cost bound for building the subtree under it decreased by $cost(\theta)$. Figure 1 exemplifies the recursive invocation of TDIDT$. The initial ρ was \$60 (left-hand figure). Since two tests (a_2 and a_5) with a total cost of \$30 have been used, their current cost is zero and the remaining budget for the TDIDT$ process is \$30. a_1, whose cost is \$40, is filtered out. Assume that, among the remaining candidates, a_6 is chosen. The subtrees of the new split (right-hand figure) are built recursively with a budget of \$20. Algorithm 1 formalizes the modified TDIDT procedure, denoted TDIDT$. TDIDT$ is designed to search in the space of trees whose test-cost is at most ρ. It can be instantiated by means of an attribute selection procedure.

Algorithm 1 TDIDT$

Input: examples E, attributes A, maximal cost ρ
if $E = \emptyset$ **then**
  return Leaf(nil)
end if
if $\exists c$ such that $\forall e \in E$ class$(e) = c$ **then**
  return Leaf(c)
end if
$\theta \leftarrow$ Choose-Test(A, E, ρ)
$V \leftarrow$ Outcomes(θ)
for all $v_i \in V$ **do**
  $E_i \leftarrow \{e \in E \mid \theta(e) = v_i\}$
  $S_i \leftarrow$ TDIDT$(E_i, A, \rho - cost(\theta))$
end for
return Node$(\theta, \{\langle v_i, S_i\rangle \mid i = 1 \ldots |V|\})$

Our first proposed anycost learner, called C4.5$, instantiates TDIDT$ by taking information gain as the split criteria, as in C4.5. Unlike the original C4.5, C4.5$ filters out unaffordable tests. It does not, however, consider the costs of the remaining candidates. Assume, for example, that the cost of the attribute with the highest gain is exactly ρ, the maximal classification budget. C4.5$ will choose to split on it and will not be able to further develop the tree.

The EG2 algorithm (Nunez, 1991) chooses attributes based on their Information Cost Function, ICF$(\theta) = (2^{\Delta I(\theta)} - 1)/(\text{cost}(\theta) + 1)$, where ΔI is the information gain. We denote by EG2$ an instantiation of TDIDT$ that uses ICF. EG2$ does not use ρ when evaluating splits and therefore might over-penalize informative yet costly attributes, even when ρ is large enough. This problem is intensified when the immediate information gain of relevant attributes is low, for example due to inter-correlation as in XOR-like concepts. Both C4.5$ and EG2$ require a fixed short runtime and cannot exploit additional resources to improve the resulting classifier.

3.2. The Pre-contract-TATA Algorithm

The recently introduced ACT algorithm overcomes the problems greedy learners encounter by allowing learn-

ing time to be traded for lower total classification costs. ACT is an anytime TDIDT algorithm that invests more time resources for making better split decisions. For every candidate split, ACT attempts to estimate the total expected cost of using the resulting subtree were the split to take place. 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 EG2, denoted SEG2. ACT is an anytime algorithm parameterized by r, the sample size.[1] When r is larger, the resulting estimations are expected to be more accurate, improving the final tree. ACT attempts to minimize the sum of the testing and misclassification costs. However, it does not consider the maximal classification budget and may violate testing cost limits. Therefore, it is inappropriate for building pre-contract classifiers. TATA has a different goal: minimizing the misclassification costs given a bound on the testing costs. Hence, TATA must take a different approach for (1) top-down induction, (2) pruning, (3) biasing the sample, and (4) evaluating trees in the sample.

While ACT adopts the TDIDT approach, in TATA we use TDIDT$. This carries 2 benefits: first, it guarantees that the produced tree will not violate the maximal cost, and second, it filters out some of the attributes and saves their evaluation time, which can be costly in the anytime setup. Once the tree is built, ACT post-prunes it in an attempt to reduce testing and misclassification costs. In TATA reducing testing costs is not beneficial because the built tree already fits the budget. Therefore, the objective of TATA's post-pruning is to reduce misclassification costs. When misclassification costs are uniform, this problem reduces to maximizing accuracy and thus we adopt C4.5's error-based pruning (EBP). When misclassification costs are not uniform, we slightly modify EPB, and prune if the expected misclassification costs decrease (rather than the expected error).

Like ACT, TATA samples the spaces of subtrees under each split to estimate its utility. ACT uses SEG2 to bias the sample towards low-cost trees. In TATA, however, we would like to bias our sample towards accurate trees that fit our testing costs budget. Therefore, instead of using SEG2, we designed a stochastic version of C4.5$, called SC4.5$. In SC4.5$ attributes are chosen semi-randomly, proportionally to their information gain. Note that the cost of the attributes affects the split decisions twice: first, TDIDT$ filters out the unaffordable tests, and second, the sampled trees themselves must fit the remaining budget, which penalizes expensive tests when the budget is relatively low.

[1] As a contract algorithm, ACT assumes predetermined learning time. Mapping contract time to r was discussed in a previous work (Esmeir & Markovitch, 2007b).

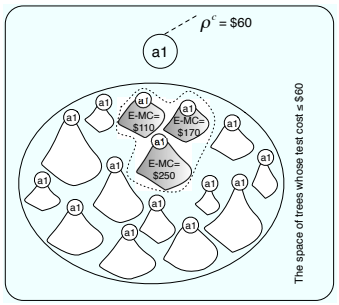

Figure 2. Attribute evaluation in Pre-Contract-TATA

Algorithm 2 TATA-Choose-Attribute

Input: examples E, attributes A, repeats r, maximal cost ρ
if $r = 0$ **then**
  return C4.5$-Choose-Attribute$(E, A, \rho)$
end if
for all $\theta \in \{\theta \in A | cost(\theta) < \rho\}$ **do**
  $V \leftarrow$ Outcomes(θ)
  for all $v_i \in V$ **do**
    $T \leftarrow$ C4.5$(E_i, A, \rho - cost(\theta))$
    $min_i \leftarrow$ ExpectedMC(T)
    for $r - 1$ **do**
      $T \leftarrow$ Stochastic-C4.5$(E_i, A, \rho - cost(\theta))$
      $min_i \leftarrow \min(min_i,$ ExpectedMC$(T))$
    end for
  end for
  $total_\theta \leftarrow \sum_{i=1}^{|V|} min_i$
end for
return θ for which $total_\theta$ is minimal

Having decided about the sampler, we need to decide how to evaluate the sampled trees. ACT favors trees whose expected total cost is lowest. In TATA all sampled trees fit the budget and therefore we choose to split on the attribute whose subtree is expected to have the lowest misclassification cost, as exemplified in Figure 2. Algorithm 2 formalizes the split selection component in pre-contract-TATA. Pre-contract-TATA is an anytime algorithm parameterized by r, the sample size. For a given set of examples and attributes, the runtime complexity of TATA grows linearly with r, just as it does in ACT (Esmeir & Markovitch, 2007a). When we cannot afford sampling ($r = 0$), TATA builds the tree using C4.5$.

Algorithm 3 Contract-TATA-Learn

Input: examples E, attributes A, repeats r, repertoire size k
$\rho_{min} \leftarrow$ Minimal-Attribute-Cost(A)
$\rho_{max} \leftarrow$ Total-Attribute-Costs(A)
$C \leftarrow \bigcup_{i=0}^{k-1}\{\rho_{min} + \frac{i}{k-1}(\rho_{max} - \rho_{min})\}$
return $\{\langle c, \text{Pre-contract-TATA}(E, A, r, c)\rangle \mid c \in C\}$

Algorithm 4 Contract-TATA-Classify

Input: repertoire R, attributes A, maximal cost ρ
$c^* \leftarrow \max\{c \mid c \le \rho \wedge \exists T, \langle c, T\rangle \in R\}$
$T^* \leftarrow T \mid \langle c^*, T\rangle \in R$
return Classify(T^*, e)

3.3. Learning Contract Classifiers

The *pre-contract* scenario assumes that ρ, the bound on testing costs, is known to the learner. In many real-life scenarios, however, we do not know ρ before building the model and thus we need classifiers that get ρ as a parameter (*contract* classification). TDIDT$-based algorithms cannot be used as are because they rely on ρ. Obviously, C4.5 and EG2 can be invoked to produce *contract* trees because they do not rely on the value of ρ. However, their performance is expected to be poor because they are not designed to control testing costs. Therefore, we are looking for a learner that has the advantages of *pre-contract*-TATA without getting the value of ρ as parameter.

A good contract classifier must be prepared to utilize any ρ it gets. We suggest using pre-contract-TATA and build a collection of k trees, each with a different value of ρ. We call such a collection a *repertoire*. In pre-contract-TATA we exploit extra learning resources in order to increase r, the sample size, and thus improve attribute-utility estimations. When composing a repertoire we can benefit from additional learning time either for building trees of higher quality or for increasing the granularity of the repertoire. The values of k and r impose a tradeoff between tree quality and granularity. Larger values of r mean better trees. Larger values of k mean less time to invest in each tree and higher memory requirements but increase the chances of finding a tree that fits the allocation. These chances depend also on the selection of ρ values. Let ρ_{max} be the cost of taking all tests and let ρ_{min} be the cost of the least expensive attribute. Obviously, there is no need to build trees with $\rho > \rho_{max}$ or $\rho < \rho_{min}$. Therefore, we uniformly distribute the k values in the range $[\rho_{min} - \rho_{max}]$, where the difference in the cost contract of each two successive trees is $\delta = \frac{\rho_{min} - \rho_{max}}{k-1}$. Algorithm 3 formalizes the procedures for forming repertoires with uniform gaps under the contract setup.

Unlike ensembles that combine predictions from several trees, the final decision of a repertoire classifier is based on a single tree. In the contract setup, classification resources are pre-allocated and therefore we pick the tree that best fits the cost bound. Given a repertoire B and a classification budget ρ, we would use the $(\frac{\rho - \rho_{min}}{\delta} + 1)^{th}$ tree in the repertoire. Algorithm 4 formalizes the procedures for using a repertoire to classify a case under the contract setup.

3.4. Learning Interruptible Classifiers

In the interruptible setup, not only is the classification budget not provided to the learner but it is not provided to the classifier. The classifier is expected to utilize resources until it is interrupted and queried for class label. To operate in such a setup we again form repertoires with uniform-gaps. Unlike the contract setup, however, ρ is not available to the classifier. Therefore, we cannot choose a single tree for classification. Instead, we start by using the lowest-cost tree and move on to the next ones, as long as resources permit. When interrupted, we use the prediction of the last fully explored tree as the final decision.

While simple, this approach raises two important questions. First, because we are charged for the cumulative cost of using the trees, it is no longer ideal to have as many trees in the repertoire as possible. Second, when forming the repertoire, the learner should take into account tests that appear in previous trees in the repertoire because their outcome might have been already known when using the tree for classification. We intend to address these questions in a future research.

4. Experimental Evaluation

A variety of experiments were conducted to test the performance and behavior of TATA in 3 scenarios: *pre-contract*, *interruptible*, and *contract* classification. Recently, (Esmeir & Markovitch, 2007a) have presented an automatic method for assigning testing costs to attributes in existing datasets. We applied this method 4 times on 20 UCI (Asuncion & Newman, 2007) problems and another 5 datasets that hide hard concepts and have been used in previous works. An online appendix gives detailed description on these datasets.[2] Following the recommendations of (Bouckaert, 2003), 10 runs of a 10-fold cross-validation experiment were conducted for each dataset and the reported results are averaged over the 100 runs.

[2] www.cs.technion.ac.il/~esaher/publications/rbc

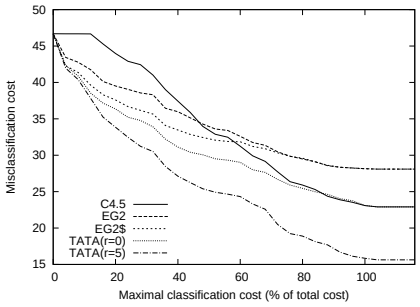

Figure 3. Pre-contract results

Table 1. Comparing the misclassification cost for different testing cost contracts

Learner	$\int_{0.33\rho_{max}}^{\rho_{max}} MC(\rho)$	TATA wins Wilcoxon
TATA($r=5$)	**21.12**	
C4.5	28.84	√
TATA($r=0$)	26.93	√
EG2	31.21	√
EG2$	30.48	√

4.1. Pre-contract Classification

Our first set of experiments compares C4.5, EG2, EG2$, TATA($r=0$), which is equivalent to C4.5$, and TATA($r=5$) in the pre-contract setup. Misclassification cost has been set uniformly to 100.[3] For each dataset we invoked the algorithms 30 times, each with a different ρ value taken from the range $[0, 120\%\rho_{max})$, with uniform steps. Figure 3 describes the misclassification cost of the different algorithms, as a function of ρ. For each point (ρ value),the results are averaged over the 100 datasets.

Clearly, TATA($r=5$) is dominant. When $\rho \leq \rho_{min}$, the algorithms cannot administer any test and thus their performance is identical. At the other end, when $\rho \geq \rho_{max}$, the attribute costs are actually not a constraint. In this case TATA($r=5$) performed best, confirming the results reported by (Esmeir & Markovitch, 2007a) when misclassification costs were dominant. The more interesting ρ values are those in between. Table 1 lists the normalized area under the misclassification cost curve over the range $[33\% - 99\%]\rho_{max}$. Confirming the curves, the results indicate that TATA($r=5$) has the best overall performance. The Wilcoxon test (Demsar, 2006), which compares classifiers over multiple datasets, finds TATA($r=5$) to be significantly better than all the other algorithms.

As expected, all five algorithms improve with the increase in ρ because they can use more features. For ρ values slightly larger than ρ_{min} we can see that EG2, which is cost-sensitive, performs better than C4.5. The reason is that EG2 takes into account attribute costs and hence will prefer lower cost attributes. With the increase in ρ and the relaxation of cost constraints, C4.5 becomes better than EG2. It is interesting to compare the TDIDT$ variants of C4.5 and EG2 to their TDIDT counterparts. It is easy to see that both TDIDT$ variants exhibit better anycost behavior, until the point where all relevant attributes can be used ($\rho \sim \rho_{max}$), where the performance of each couple becomes identical. The advantage of the TDIDT$ variants is because they will not choose tests that violate the cost limits and therefore will not be forced to stop the induction process earlier.

A comparison of TATA($r=0$) to TATA($r=5$) indicates that the latter is clearly better: while TATA($r=0$) chooses split attributes greedily, TATA($r=5$) samples the possible subtrees of each candidate and bases its decision on the quality of the sampled trees. Of course, the advantage of TATA($r=5$) comes at a price: it uses far more learning resources.

Figure 4 gives the results for 4 individual datasets, namely Glass, AND-OR, MULTI-XOR and KRK. In all 4 cases TATA is dominant with its $r=5$ version being the best method. We can see that the misclassification cost decreases almost monotonically. The curves, however, are less smooth than the average result from Figure 3 and slight degradations are observed. The reason could be that irrelevant features become available and mislead the learner. The KRK curve looks much like steps. The algorithms improve at certain points. These points represent the budgets when the use of another relevant attribute becomes possible. The graphs of AND-OR and MULTI-XOR do not share this phenomenon because these concepts hide interdependency between the attributes. As a results, an attribute may be useless if other attributes cannot be considered. The performance of the greedy C4.5 and TATA($r=0$) on these problems is very poor.

Besides being able to produce good anycost trees, TATA itself is an anytime algorithm that can trade learning resources for producing better trees. Our next experiment examines the anytime behavior of TATA by invoking it with different values of r. Figure 5 shows the results. As we can see, all TATA versions are better than C4.5. With the increase in r, the advantage of TATA increases. The most significant improvement is

[3] Note that the absolute value of the misclassification cost does not matter because we do not assume same-scale.

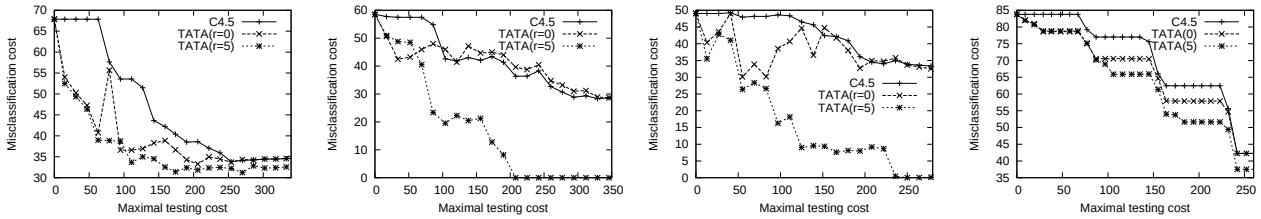

Figure 4. Pre-contract results: the misclassification cost as a function of the preallocated testing costs contract for one instance of Glass (left), AND-OR, MULTI-XOR, and KRK

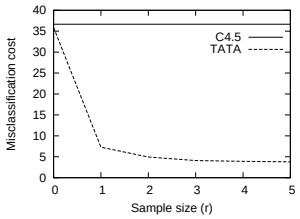

Figure 5. TATA with different sample sizes on the Multi-XOR dataset

from $r = 0$ to $r = 1$. TATA with $r = 0$ is a greedy algorithm that uses local heuristics. When $r > 0$, TATA uses sampling techniques to evaluate attributes and therefore becomes significantly better. For more difficult concepts, where only a combination of a large number of attributes yields information, a larger value of r would be needed.

4.2. Contract and Interruptible Classification

TATA uses repertoires to operate in the contract setup. To examine the anycost behavior of TATA in the contract setup we built 2 repertoires, each of size 16. The first repertoire uses *pre-contract*-TATA($r = 0$), which is equivalent to C4.5\$, to build the trees. The second repertoire uses *pre-contract*-TATA($r = 3$). The repertoires were used to classify examples in the contract setup. 120 uniformly distributed ρ values in the range $[0-120\%\rho_{max}]$ were used as contract parameters. Figure 6 describes the performance of these three repertoires on 4 datasets (Glass, AND-OR, MULTI-XOR, and KRK), averaged over the 100 runs of 10 times 10-fold cross-validation. In addition, we report the results of the cost-insensitive C4.5. It is easy to see that across all 4 domains TATA($r = 3$) repertoires are dominant. TATA($r = 0$) repertoires are better than C4.5 when the provided contracts are low. When the contracts can afford to use all the attributes, both algorithms perform similarly. In comparison to TATA($r = 0$), the anycost behavior of TATA($r = 3$) is better: it is monotonic and utilizes testing resources better.

In the interruptible classification setup, TATA forms a repertoire of trees and traverses it until interrupted. Unlike the contract setup, here we would like to limit the number of trees in the repertoire to avoid wasting resources on tests that will not be needed by the tree that makes the final decision. This would be the case even if we had infinite learning resources. However, too small repertoires may lead to poor performance in significant ρ ranges. To examine the anycost behavior of TATA in the interruptible setup, we built different TATA repertoires that were used to classify examples in the interruptible setup. Like the *contract* case, the TATA repertoires achieved the best performance along a wide range of ρ values.

5. Related Work and Conclusions

Many applications limit the resources that can be used for classification, and therefore the classifiers must be able to operate under strong testing cost constraints. In this work we described 3 different scenarios for anycost classification. In the *pre-contract* setup the maximal cost is known to the learner, in the *contract* setup it is known after building the predictive model but before classification, and in the *interruptible* the limit is not predetermined. The major contribution of this paper is the TATA framework which is designed to operate in such environments.

For the *pre-contract* setup, TATA builds a tree top-down and uses sampling techniques to make better split decisions. For the *contract* and *interruptible* scenarios, TATA builds a repertoire of trees, formed from several invocations of *pre-contract*-TATA with different cost limits. TATA's components are anytime and therefore can exploit extra learning time to build better anycost classifiers. The experimental study shows that TATA exhibits good anytime behavior, and produces significantly better anycost classifiers.

While, to the best of our knowledge, there has been no specific attempt to design an anytime algorithm for learning anycost decision trees, there are several

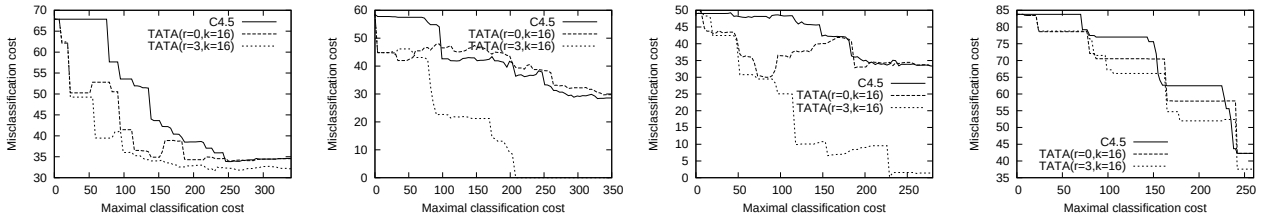

Figure 6. Contract results: the misclassification cost as a function of the preallocated testing costs contract for Glass (left), AND-OR, MULTI-XOR, and KRK

related works, aside from those mentioned in the previous sections, that warrant discussion here.

Yang et al. (2007) introduced Anytime Averaged Probabilistic Estimators (AAPE) for utilizing additional computational recourses during classification. At classification time, AAPE computes Naive-Bayes and then exploits extra time to refine the probability estimate. Unlike TATA, AAPE is a fixed-time learner. Moreover, AAPE does not deal with attribute costs and assumes that the major constraint is the computation time needed by the probabilistic models.

Sheng et al. (2006) addresses the related problem of test strategies: given a tree, which values to query for and in what order. Several test strategies have been studied, including sequential, single batch and multiple batch These strategies are orthogonal to our work because they assume a tree is already built.

TATA makes it possible for many real-life problems with testing cost constraints to use decision-trees. In the future we intend to examine other methods for producing the TATA samples. In addition, we plan to research the tradeoff between the size of a *repertoire* and the quality of the trees composing it.

References

Asuncion, A. and Newman, D.J. UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences, 2007. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

Bilgic, Mustafa and Getoor, Lise. Voila: Efficient feature-value acquisition for classification. In *AAAI'07*, July 2007.

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

Bouckaert, R. R. Choosing between two learning algorithms based on calibrated tests. In *ICML'03*, 2003.

Demsar, J. Statistical comparisons of classifiers over multiple data sets. *JMLR*, 7:1–30, 2006.

Esmeir, S. and Markovitch, S. Anytime induction of cost-sensitive trees. In *Proceedings of NIPS'07*, 2007a.

Esmeir, S. and Markovitch, S. Anytime learning of decision trees. *JMLR*, 8, 2007b.

Greiner, R., Grove, A. J., and Roth, D. Learning cost-sensitive active classifiers. *Artificial Intelligence*, 139 (2):137–174, 2002. ISSN 0004-3702.

Kapoor, A. and Greiner, R. Learning and classifying under hard budgets. In *Proceedings of ECML'05*, 2005.

Lizotte, Daniel J., Madani, Omid, and Greiner, Russell. Budgeted learning of naive bayes classifiers. In *UAI'03*, 2003.

Melville, P., Saar-Tsechansky, M., Provost, F., and Mooney, R. J. Active feature acquisition for classifier induction. In *ICDM'04*, 2004.

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

Sheng, S., Ling, C. X., Ni, A., and Zhang, S. Cost-sensitive test strategies. In *AAAI'06*, 2006.

Sheng, V. S. and Ling, C. X. Partial example acquisition in cost-sensitive learning. In *KDD'07*, 2007.

Turney, P. Types of cost in inductive concept learning. In *Proceedings of the Workshop on Cost-Sensitive Learning held with ICML'00*, 2000.

Turney, Peter D. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *JAIR*, 2:369–409, 1995.

Yang, Y., Webb, G., Korb, K., and Ting, K. Classifying under computational resource constraints: anytime classication using probabilistic estimators. *Machine Learning*, 69:35–53, 2007.