Budgeted Learning of Bounded Active Classifiers

Aloak Kapoor
aloak@cs.ualberta.ca

Russell Greiner
greiner@cs.ualberta.ca

Department of Computing Science
University of Alberta
Edmonton, AB T6J 2E8

ABSTRACT

Since resources for data acquisition are seldom infinite, both learners and classifiers must act intelligently under hard budgets. In this paper, we consider problems in which feature values are unknown to both the learner and classifier, but can be acquired at a cost. Our goal is a learner that spends its fixed learning budget b_L acquiring training data, to produce the most accurate "active classifier" that spends at most b_C per instance. To produce this fixed-budget classifier, the fixed-budget learner needs to sequentially decide which feature values to collect in order to learn the relevant information about the underlying distribution. We explore a variety of approaches the learner can take, including the standard "round robin" policy (purchasing every feature of every instance until the b_L budget is exhausted). In this work, we demonstrate empirically that the round robin strategy is problematic (especially for small b_L), and provide alternate learning strategies that achieve superior performance on a variety of real-world datasets.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Induction, Knowledge acquisition, Parameter learning*

General Terms

Algorithms

Keywords

Active classifiers, budgeted learning, classification costs, data acquisition, learning costs

1. INTRODUCTION

While a doctor may have the option of using a wide variety of medical tests (including MRIs, blood work, etc.) to diagnose a patient, many medical plans involve capitation payments that restrict the per-patient cost of medical diagnosis and treatment. These physicians can only consider diagnostic strategies that spend at most a specified amount; they would clearly want to use the most accurate such strategy. In general, these strategies can operate sequentially: e.g. first performing test Blood_7 (at cost $C(\text{Blood}_7)$), then using this information to decide on the next action; perhaps performing Liver_3 if Blood_7 was positive, but performing Urine_2 if Blood_7 was negative, and so forth. Once the total cost of the tests performed reaches the capitation amount b_C (i.e. if $C(\text{Blood}_7) + C(\text{Urine}_2) + \cdots = b_C$), the strategy must stop collecting information and render a decision — e.g. "Cancer = true". We call such a strategy a "bounded active classifier" [6].

Earlier results [6] have shown that one can PAC-learn this optimal "bounded active classifier" $\text{BAC}^* = \arg\min_b \{\text{error}(b) | b \in \text{cost-}b_C\text{-active classifiers}\}$, assuming the *learner* has no *a priori* resource bound — i.e. it can purchase every feature of as many instances as necessary. Of course, if we are charging the classifier (read "physician") for each feature, it seems strange to provide this information for free to the learner (think "experimental designer"). This paper extends those earlier results by investigating the challenge of learning this BAC^* when the *learner* has a fixed budget to spend acquiring the relevant training data — i.e., when the learner can spend only a total of b_L to produce the best classifier that can spend only b_C per instance. Thus, we investigate the problem of budgeted learning of a bounded active classifier.

Although the task is NP-hard in general [6], we demonstrate how to improve the running time of the optimal algorithm, propose a variety of tractable learning strategies, and run tests on datasets with features of varying predictive power. Our learning strategies are able to beat the obvious round robin approach (which spends equally on all features) by significant margins.

The rest of this paper is organized as follows. Section 2 formally introduces the framework for budgeted learning a bounded active classifier, highlights the simplifying assumptions we make, and derives complexity results. Section 3 discusses a variety of algorithms that attempt to find good approximate solutions to the problem, including the obvious round robin purchasing as well as novel approaches. Section 4 gives empirical results that compare the proposed algorithms and discusses their effectiveness. Section 5 reviews related literature from machine learning and statistics, and Section 6 summarizes contributions and considers future work. All proofs are included in the Appendix.

UBDM August 24, 2005, Chicago, Illinois, USA

2. FORMAL DESCRIPTION

The "budgeted bounded-active-classifier learner", bBACL, is given the (non-negative) cost $C(X_i) \in \Re^+$ of acquiring each individual feature X_i of any single specified instance[1] and the loss matrix $L = [\ell_{i,j}]$ whose (i, j) element specifies the penalty for returning the class c_i when the true class is c_j; by convention we assume $\ell_{i,i} = 0$ and $\ell_{i,j} > 0$ for $i \neq j$. bBACL also knows the total amount the learner can spend $b_L \in \Re^+$, and how much the resulting active classifier can spend per instance $b_C \in \Re^+$.

Throughout, the bBACL can see the current $m \times (r + 1)$ "tableau", whose rows each correspond to an instance $i \in \{1, \ldots, m\}$ and whose first r columns each correspond to a feature, and whose $r+1$st column is the class label. Initially, only the class label is specified; the other $m \times r$ entries are all unknown. In general, we will let $x_i^{(j)}$ refer to the initially unknown value of the ith feature of the jth instance. At any point, bBACL can perform the (i, j) "probe" to determine the value of $x_i^{(j)}$, at cost $C(X_i)$. This also reduces bBACL's remaining budget from b_L to $b_L - C(X_i)$. Once this budget reaches zero, bBACL stops collecting information and returns a bounded active classifier BAC, which corresponds to a decision tree of bounded depth [4].

The score of any BAC B is its expected misclassification error:

$$Q(B) \quad = \quad \sum_{\mathbf{x},y} P(\mathbf{x}, y)\ L(B(\mathbf{x}), y) \qquad (1)$$

Letting $All(b_C)$ be the set of all such active classifiers that spend at most b_C per instance, our goal is the BAC from this set that minimizes this error:

$$\text{BAC}^* = \underset{B \in All(b_C)}{\arg\min}\ Q(B) \qquad (2)$$

2.1 Simplifying Assumptions

For our work we will assume a constant misclassification cost $\ell_{ij} = 1$ for $i \neq j$ and $\ell_{ii} = 0$. Our algorithms will need to estimate the probabilities over the values of the features of an instance $P(x_i^{(j)})$ to decide which probe to perform. We will take a Bayesian stance by assuming there is a prior distribution over labeled instances, before seeing any data.[2] As a simplification, we will make the Naïve Bayes assumption, which means the distribution of $x_i^{(j)}$ is independent of $x_k^{(j)}$ (for $k \neq i$) as we know the value of the class y_j.[3] Hence, if instance j is labeled with class $+$, we will model the distribution of its ith feature $x_i^{(j)} \sim \text{Dir}(\,\alpha_{1,+}^{(i)}, \ldots, \alpha_{w,+}^{(i)}\,)$ as a Dirichlet distribution with parameters $\alpha_{j,+}^{(i)} > 0$, assuming X_i has w values [7]. These parameters are unrelated to the ones for negatively labeled instances $\alpha_{j,-}^{(i)}$ and also unrelated to the parameter values for other features $X_{h \neq i}$. Initially, we will assume that each such distribution is uniform Dir(1, . . . , 1). If we later see a sample S with 29 $Y = +$ instances with $X_i = +$ and 14 $Y = +$ instances with $X_i = -$, the posterior distribution for $x_i^{(j)}$ for a new $Y = +$ instance would be Dir($1 + 29,\ 1 + 14$). The mean probability for $X_i = +$ here would be $P(X_i = +|S) = 30/(30 + 15) = 2/3$.

In general, if a variable X's prior distribution is $X \sim \text{Dir}(\,\alpha_1, \ldots, \alpha_w\,)$, then

$$P(X = i) \quad = \quad \frac{\alpha_i}{\sum_k \alpha_k} \qquad (3)$$

If we then observe a sample S that includes a_i instances of $X = i$, then X's posterior distribution remains a Dirichlet, with new parameters

$$X|S \quad \sim \quad \text{Dir}(\,\alpha_1 + a_1, \ldots, \alpha_w + a_w\,) \qquad (4)$$

In the formal description above, a probe of the form $x_i^{(j)}$ specifies the feature to probe (X_i) and the specific instance in the tableau (instance j) on which to perform the probe. However, because of our Naïve Bayes assumption, we can treat all instances with the same class label identically. Thus, rather than querying specific instances, we only consider probes of the form (i, y) that request the ith feature of a randomly chosen instance in the tableau whose class label is y. (By convention, this process selects the value of an (i, y) instance that has not been seen before.)

2.2 Complexity Results

[11] proves the following much simpler task is NP-hard: Given a set of coins with known prior distributions and a fixed total number of flips, decide when to flip which coin to decide which coin has the highest head probability. Our framework inherits that negative result. (Identify each coin f_i with a binary feature, whose head probability corresponds to the probability the class is true, given f_i is true, $P(c = +|f_i = +)$; we also let $P(c = +|f_i = -) = 0$ for all features.) In addition, [6] shows that computing the best active classifier is NP-hard in general, even if we know the entire distribution. Again, our framework inherits that negative result. That paper also provided a tractable algorithm for (PAC-)learning a *bounded* active classifier, after observing a polynomial number of complete instances; this observation motivates our interest in finding such optimal bounded active classifiers.

3. ALGORITHMS

This section summarizes a number of "budgeted bounded-active-classifier learners". We focus on only the data collection part of the algorithms; after collecting \$$b_L$ worth of feature-values, each of the algorithms then passes its learned (posterior) Dirichlet distributions to a dynamic program that produces the BAC* in Equation 2.

3.1 Optimal Policy

As our problem is a finite Markov Decision Process, there exists a deterministic optimal policy for spending the learning budget such that the expected$_1$ total (expected$_2$) misclassification error[4] of the final bounded active classifier is

[1]We assume that these costs are independent of each other, both within and across instances. Moreover, if any test costs $C(X_i) = 0$, we can simply gather that information for each instance and consider the remaining reduced problem where $C(X_i) > 0$ for all remaining X_is.

[2]The sparsity of the data means the obvious frequentist approach of using simple frequencies is problematic.

[3]Note that Naïve Bayes models often produce good classifiers even for datasets that violate this assumption.

[4]The first expectation$_1$ is over the set of possible Dirichlet distributions produced by the learner's purchases, and the

minimized. Mathematically, the optimal learning policy is the one that minimizes Equation 5:

$$\sum_{i \in Outcomes} P(i) \sum_{\mathbf{x},y} P(\mathbf{x},y|i)\ L(BAC^*(\mathbf{x}),y) \qquad (5)$$

where each "outcome" corresponds to a state in which our learning budget has been fully exhausted and has resulted in posterior Dirichlet distributions over the feature values.

Such a policy can be computed via a bottom-up dynamic program. To see this, note that we can compute the value of all possible outcomes where the learning budget has been exhausted, and then use these to compute the value of all possible outcomes where there is only \$1 left in the learning budget, and so on. Unfortunately, the number of outcomes (and hence the computational complexity) has a prohibitive lower bound:

PROPOSITION 1. *Let $|X_i|$ denote the domain size of feature X_i, $|S|$ denote the number of classes, $t = |S| \sum_i |X_i| - 1$, and each feature has unit cost. Then the bottom-up dynamic program must compute the value of* $\Omega\left(\left(\frac{b_L+t}{b_L}\right)^{b_L}\left(\frac{b_L+t}{t}\right)^{t}\frac{1}{\sqrt{t}}\right)$ *outcomes.*

Given the exponential dependence on the learning budget and the domain sizes of all the features, the straightforward calculation of the optimal policy via dynamic programming can tractably solve only small problems.

We have considered methods of improving upon this naïve dynamic program, namely by reducing the number of subproblems that must be solved. Below we show an interesting way to achieve this reduction by exploiting the equivalence of two "permuted" states under the conditional independence assumption.

Definition 1. A *proper permutation* for a feature X_i with t domain values is a bijective function $f : [1,t] \to [1,t]$ that reorders the t parameters for every Dirichlet distribution on X_i.

Example 1. Let

$$(X_i|Y=0) \sim \mathrm{Dir}(4,2,7), \quad (X_i|Y=1) \sim \mathrm{Dir}(3,8,5)$$

Then a proper permutation for feature X_i is:

$$(X_i|Y=0) \sim \mathrm{Dir}(7,2,4), \quad (X_i|Y=1) \sim \mathrm{Dir}(5,8,3).$$

PROPOSITION 2. *Let the Naïve Bayes assumption hold, and let us identify a "state" of our problem by the value of b_L and the set of Dirichlets over the feature-class pairs. Consider any two states A and B, that have equal values of b_L and are such that the Dirichlets of A can be made equal to the Dirichlets of B by specifying a set of r proper permutations, one for each feature X_i. Under these conditions, the expected value of state A is equal to the expected value of state B when following an optimal policy, and the optimal action to take from state A is the optimal action to take from state B.*

second expectation$_2$ is over the possible labelled instances $(\mathbf{x}, y)$ that can occur *given* the resulting Dirichlets

Table 1: Reduction in computation time using Proposition 2

b_L	b_C	Features	Domain Size	Naïve	Improved
2	4	6	4	161 sec	65 sec
3	2	4	3	888 sec	432 sec
4	3	4	3	8280 sec	3360 sec

This proposition allows us to improve the naïve dynamic program by reusing the computed value of a state A for properly permuted versions of A. The real-time improvement using Proposition 2 is shown in Table 3.1. In the last case, the naïve dynamic program ran out of memory after more than two hours, while our improved version finished properly in under an hour. Unfortunately such improvements are not sufficient to remove the exponential complexity of the dynamic program, (recall that this task is NP-complete) leading to the more tractable, suboptimal approaches we consider next.

3.2 Round Robin (RR)

This obvious algorithm simply purchases *complete* instances until its budget b_L is exhausted. It draws examples randomly, and so expects to have collected data about members of each class y in proportion to $P(Y=y)$. If there are r unit-cost features, we expect to know everything about roughly b_L/r instances. Notice this approach implicitly assumes that all features are equally valuable in learning the target concept.

3.3 Biased Robin (BR)

A more selective approach than Round Robin is to purchase a single feature and test whether or not its observed value has increased some measure of quality. The Biased Robin algorithm is more selective than RR, continually purchasing feature X_i as long as it improves quality, and otherwise moving to feature X_{i+1} (and of course looping back to X_1 after X_r). There are several choices for how to measure quality or loss; see Section 3.6. As further motivation for this algorithm, [10] found it to be one of the best approaches for budgeted learning of a passive Naïve Bayes classifier, albeit with a different loss function. This method also corresponds to the "Play the Winner" approach discussed in [16].

3.4 Single Feature Lookahead (SFL)

One would always like to avoid wasting purchases on poor features, especially when faced with a limited learning budget. This motivates a prediction-based approach, which uses a loss function to approximately predict the expected loss incurred after making a sequence of purchases of the identified feature. If only the next one purchase is considered, then this reduces to the (1-step) greedy algorithm.

SFL uses this prediction based approach, but controls the level of myopia or "greediness" involved by providing an additional parameter, d = the lookahead depth. With a lookahead depth of d, SFL calculates the expected loss of spending its next \$$d$ sequentially purchasing feature i of instances of class j. That is, if S denotes our current set of Dirichlets and S' denotes the Dirichlets after spending $\min(\$d, \$b_L)$ purchasing feature X_i of $Y = j$ instances, then the expected loss for (i, j) is:

$$SFL(i,j) \quad = \quad \sum_{S'} P(S'|S)\ Loss(S') \qquad (6)$$

SFL determines the feature-class pair (i, j) with lowest expected loss, then purchases the value of this best (i, j) feature for *one* instance, and updates the Dirichlets based on the observed outcome of that purchase (and reduces the available remaining budget). It then recurs, using Equation 6 to compute the score for all feature-class pairs in this new situation — with its updated Dirichlets and a smaller budget. This process repeats until the learning budget is exhausted. The lookahead depth d can be set based on the computational resources available. For datasets with many features and a large learning budget, full lookahead (i.e., $d = b_L$) may be too expensive to perform, and thus d can be set to a more shallow, tractable level. SFL was originally used in two previously investigated variants of the budgeted learning problem [10, 12].

3.5 Randomized SFL (RSFL)

Our experiments show that the SFL algorithm often spends the majority of its probes purchasing a single discriminative feature-class pair and neglects to explore other potentially good features. This property can be problematic, particularly when a dataset contains several discriminative features that can jointly yield a more accurate BAC than any single feature by itself. The Randomized Single Feature Lookahead algorithm (RSFL) alleviates this problem by increasing exploration among the best looking feature-class pairs. The RSFL algorithm is very similar to SFL, as it too calculates the expected loss in Equation 6 for each feature-class pair. However, rather than deterministically purchasing the pair with the best SFL score, RSFL considers the best K feature-class pairs and for each feature-class pair (i, j) in this set, it chooses to purchase feature i of class j with probability:

$$\frac{\exp \frac{-SFL(i,j)}{\tau}}{\sum_{i,j} \exp \frac{-SFL(i,j)}{\tau}} \qquad (7)$$

Here τ is a temperature controlling exploration versus exploitation, and is set to 1 throughout this paper. After experimenting with various values for the number of feature-class pairs, K, we found that $K =$ (number of classes) $\times$ (b_c) worked well. This parameter setting seemed to yield particularly good results when the learning budget was not much greater than the number of features (i.e., $b_L = r \times v$ with v a small positive integer).

3.6 Loss Functions

As mentioned earlier, several of our algorithms rely on a loss function

$$Loss : \{\text{Dirichlet distributions over features}\} \rightarrow \Re \qquad (8)$$

that attempts to measure the quality of a given probability distribution. After experimenting with several different choices of loss functions, we found Conditional Entropy Loss and Depth 1 BAC Loss to be effective.[5]

Conditional Entropy measures the uncertainty of the class label Y given the value of a feature X_i:

$$-\sum_x P(X_i = x) \sum_y P(Y = y|X_i = x) \log_2 P(Y = y|X_i = x) \qquad (9)$$

The Biased Robin algorithm uses Equation 9 before and after the purchase of feature X_i to determine whether the purchase improved the ability of X_i to predict the class Y.

On the other hand, other algorithms (SFL, RSFL, and greedy) use

$$\min_i \sum_x P(X_i = x) \min_y (1 - P(Y = y|X_i = x)) \qquad (10)$$

which calculates the expected misclassification error of the best Depth 1 BAC.

4. EXPERIMENTAL RESULTS

To compare the algorithms, we tested their performance on several datasets from the UCI Machine Learning Repository [19]. We used supervised entropy discretization [5] to discretize datasets with continuous values. Each dataset was then randomly partitioned into five folds. The algorithms were run five times, and on each run a single fold was set aside for testing, while the remaining four were available for purchasing. For each algorithm, we used the average value of these five runs as the algorithm's misclassification error on the whole dataset. We repeated this process 50 times to reduce the variance and get a measure of the average misclassification error. Thus, each point in the graphs that follow represents 50 repetitions of five-fold cross validation.

In the first set of experiments, all features have unit cost and the datasets contain some irrelevant features. We set the classifier's budget to $b_c = 3$, as this is large enough to allow several features to be used, but small enough to keep computations tractable. All Dirichlets parameters are uniformly initialized to 1. For reference, each graph also includes a gold standard "All Data" algorithm, which is allowed to see the *entire* dataset, and thus represents the best that one can do using the Naïve Bayes assumption on the data.

Figure 1 shows the performance of the algorithms on the Glass Identification dataset: a binary class problem with nine features whose domain sizes vary between one and three. The four features that have a domain size of one represent irrelevant information that any learning algorithm (especially one under a constraining budget) should avoid. Both RSFL and BR learn better than the obvious RR algorithm for all learning budgets considered. In fact, we found the optimal $b_C = 3$ BAC produced by the "All Data" algorithm involves four different features, and these four features are precisely

[5]The obvious loss function is just to use Equation 2 to compute the expected error of the optimal BAC. However, since loss functions can be called several times on a single purchase, the computational expense of computing Equation 2 is prohibitive.

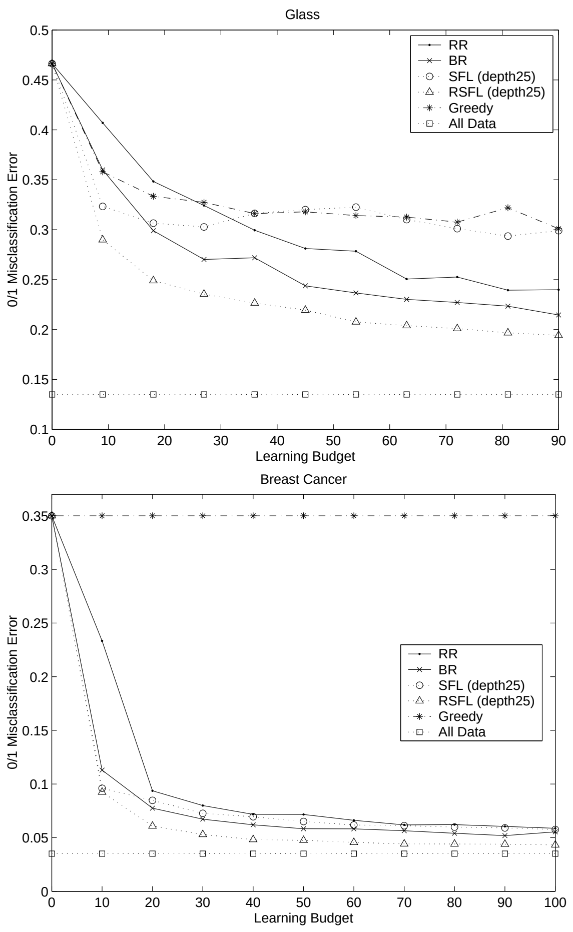


Figure 1: Identical costs and some irrelevant features — RSFL and BR outperform RR.

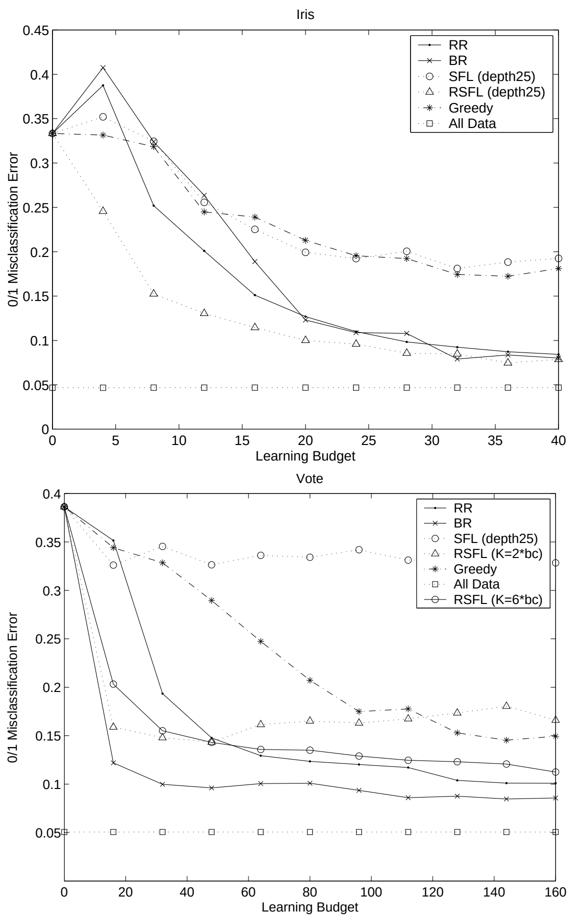


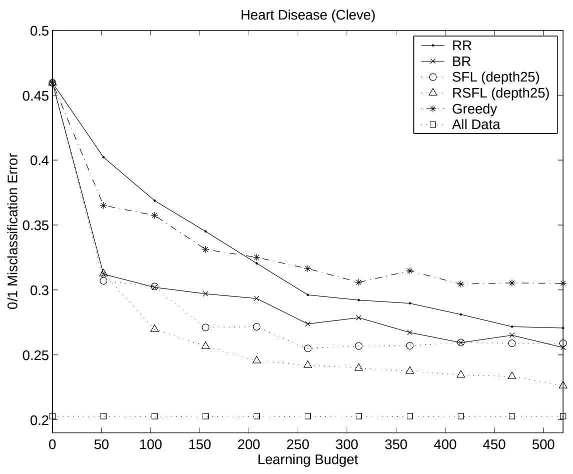
Figure 2: Identical costs, no irrelevant features — RR still suboptimal.

the ones that RSFL and BR purchase heavily during learning. This is in contrast to the RR purchasing behaviour that spends equally on all features, despite their unequal predictive power. Finally, SFL and Greedy spend their entire budget on only one or two features during learning, which accounts for their low accuracy BACs.

The Breast Cancer dataset contains ten features, only one of which is irrelevant to the concept. This dataset is particularly interesting because nearly all its features are good predictors, but three features have markedly lower conditional entropy than the rest. To produce the lowest error BAC, the learning algorithms must discover the superiority of these three features. We find RSFL does exactly this, spending 20%, 21%, and 32% of its budget respectively on the three strong features. In comparison, RR spends 10% of its budget on every feature which makes it much more difficult for it to separate the top features from the rest. BR also performs better than RR for all learning budgets considered.

The next set of experiments, shown in Figure 2, considers datasets without any irrelevant features. The Iris dataset has only four features and is a three class problem. Given that all four features are relevant, and that $b_C = 3$ in this experiment, the optimal BAC requests every feature at some point in its tree. With only four features to consider, RSFL is able to test them all effectively and produce better BACs than RR for all budgets considered. BR is also competitive with RR, except at some of the very low budgets where BR's exploration model prevents it from ever investigating some of the features.

Figure 2 (lower) shows another binary class problem, the Vote dataset, that contains 16 features. Many of these features have similar (high) predictive power. Once again we see that both RSFL and BR beat RR when the learning budget is small. RSFL asymptotes after about 50 purchases — it spends its budget finding a few strong features quickly and outputs a fairly low error BAC. As expected, at larger budgets RR collects enough information on every feature to find many more suitable candidates for its BAC than RSFL can. The graph shows that one can improve the performance of RSFL by increasing the number of top feature-class pairs that RSFL considers on this dataset. We also observe that BR's exploration model is particularly well suited to this task because it is able to collect information on every feature at larger budgets, which is crucial on a dataset such as Vote with a large number of predictive features.



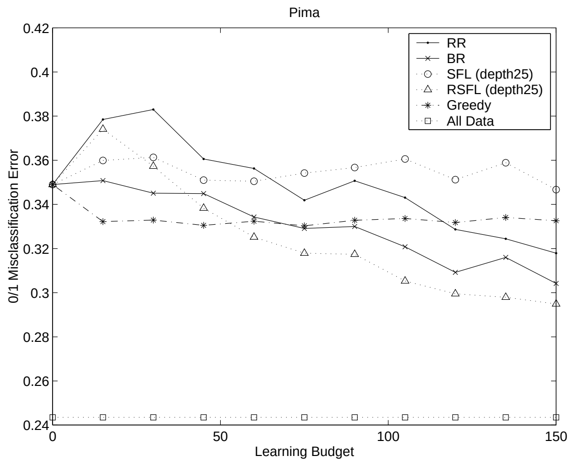


Figure 3: Different feature costs — RSFL and BR dominate RR.

Our final set of experiments involved datasets where the features differed in cost. Both the Heart Disease dataset and the Pima Indians dataset have known cost data [19], which we used in our tests. The scaled Heart Disease costs range from \$1 to \$7, and our tests are run with $b_c = \$7$. This dataset represents the worst case for RR, because the irrelevant features happen to be the most expensive ones. In fact, RSFL achieves the same error rate after \$100 that RR takes \$500 to reach. In the Pima dataset, feature costs are between \$1 and \$5, and we set $b_c = \$5$. The two irrelevant features have cost \$1, and the single best feature is \$4. Once again, BR and RSFL dominate RR for all budgets considered.

5. RELATED WORK

There are a number of different senses of "costs" in the context of learning [23]. Our research considers two of these: the costs paid by the learner to acquire the relevant information at training time to produce an effective classifier and also the costs paid by the classifier, at performance time, to acquire relevant information about the current instance. We impose hard constraints on the total cost of tests that can be performed per instance, and on the expenses paid by the learner.

Many existing (sub)fields, such as active learning [3] and experimental design [2] (as well as earlier results such as [10]) focus on only the first of these costs – e.g., bounding how much the learner can spend to produce an accurate *passive* classifier (which is not able to acquire the values of unspecified features). In addition, many of these systems request the *class label* for an otherwise *completely specified instance.* Thus they require only a single quantity per instance. Our problem is the complement of this: class labels are known but feature information must be purchased. Unlike most of the other models, this means our work may need to consider the correlations amongst the many unknown properties of an instance. Other results seeking to reduce the sample complexity for learning include decision theoretic subsampling [14], on-line stopping rules [20], progressive sampling [15], and active feature value acquisition [13]. We note that these techniques differ from our approach because we place a firm budget on the learner's ability to acquire information, while these approaches typically allow the learner to purchase until some external stopping criteria (for instance, accuracy) is satisfied.

Weiss and Provost [24] recently explored a problem related to one that we encounter in our overall framework: how to represent the class distribution when only a firm budget of n training examples can be used. For example, if our budget allows for ten training examples, should we select five from class one and five from class two, or skew our samples toward the majority class.

As for the costs paid by the classifier at performance time, both [22] and [6] attempt to produce a decision tree that minimizes expected total cost. However, neither work assumes an a priori resource bound on the learner, thereby allowing for unconstrained amounts of training data with which to build these classifiers. Again, our work makes the more realistic assumption that if data costs money at performance time, it very likely costs money at learning time as well.

Finally, we can view our model as a (fixed horizon, partially observable) Markov Decision Process (MDP) [18]: after performing a set of tests, the learner is in a "state" associated with observing the outcomes of those tests; it can then select some new test to run, which stochastically maps that state to a new state (depending on the outcome of that test). There is a vast literature on finding optimal policies for such MDPs [21]. We note that although the MDP formulation is theoretically clear, it has not yielded strong results in our experiments due to the dimensionality and lack of suitable features for function approximation; see [8]. Recent research has provided new ideas for handling large MDPs [9], but such methods are still unsuitable to our task because of the exponential dependence on the size of the learning budget (i.e., the horizon). A simpler version of our problem also exists in the MDP framework [12], and the results of that work motivate several of the policies that we adapt for budgeted learning of bounded active classifiers.

6. CONCLUSIONS

We are currently pursuing several extensions to this work. First, we are continuing to develop other tractable approximation algorithms; seeking ones with guarantees on learning performance. We are also using algorithms that go beyond the "Naïve Bayes" assumption, to allow the learner to perform more powerful probes (e.g., request feature X_i on an instance where $X_j = +$ and $Y = -$). An additional goal is to extend our learners to perform well when the misclassification costs are not uniform.

Contributions: Many standard learning algorithms implicitly assume the features are always available for free, to both the learner at "training time" and later the classifier, at "performance time". This paper extends those systems by explicitly considering these costs, at both training and performance time. It introduces the formal framework for budgeted learning a bounded active classifier, and presents some complexity results. We also propose a more efficient way to implement the optimal algorithm, which we prove works effectively. Moreover, this paper motivates and defines a variety of tractable learning strategies and shows they work effectively on various types of data — both with identical and with different feature costs. In particular, we demonstrated that our proposed strategies can often do much better than the obvious algorithm – "Round Robin" – especially when training data is limited.

Acknowledgments

Both authors wish to thank NSERC and iCORE for their generous support. We also thank Dan Lizotte and Omid Madani for insights on various related problems, and the anonymous reviewers for their comments. RG also thanks the Alberta Ingenuity Centre for Machine Learning.

7. REFERENCES

[1] G. E. Andrews. The theory of partitions. In *Encyclopedia of Mathematics and its Applications.* Addison-Wesley, 1976.

[2] K. Chaloner and I. Verdinelli. Bayesian experimental design: A review. *Statistical Science*, 1995.

[3] D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. In *Advances in Neural Information Processing Systems*, 1995.

[4] D. Dobkin, D. Gunopoulos, and S. Kasif. Computing optimal shallow decision trees. In *International Workshop on Mathematics in Artificial Intelligence*, 1996.

[5] U. Fayyad and K. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *International Joint Conference on Artificial Intelligence*, 1993.

[6] R. Greiner, A. J. Grove, and D. Roth. Learning cost-sensitive active classifiers. *Artificial Intelligence*, 2002.

[7] D. Heckerman. A tutorial on learning in bayesian networks. In *Learning in Graphical Models.* The MIT Press, 1999.

[8] A. Kapoor and R. Greiner. Reinforcement learning for active model selection. In *Utility-Based Data Mining Workshop (KDD)*, 2005.

[9] M. Kearns, Y. Mansour, and A. Y. Ng. A sparse sampling algorithm for near-optimal planning in large markov decision processes. *Machine Learning*, 2002.

[10] D. J. Lizotte, O. Madani, and R. Greiner. Budgeted learning of naive-bayes classifiers. In *Proceedings of Uncertainty In Artificial Intelligence*, 2003.

[11] O. Madani, D. J. Lizotte, and R. Greiner. Active model selection. Technical report, University of Alberta, 2004.

[12] O. Madani, D. J. Lizotte, and R. Greiner. Active model selection. In *Proceedings of Uncertainty in Artificial Intelligence*, 2004.

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

[14] R. Musick, J. Catlett, and S. Russell. Decision theoretic subsampling for induction on large databases. In *International Conference on Machine Learning*, 1993.

[15] F. Provost, D. Jensen, and T. Oates. Efficient progressive sampling. In *International Knowledge Discovery and Data Mining Conference*, 1999.

[16] H. Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 1952.

[17] S. Ross. *A First Course in Probability.* Prentice Hall, 1997.

[18] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 2002.

[19] C. B. S. Hettich and C. Merz. UCI repository of machine learning databases, 1998.

[20] D. Schuurmans and R. Greiner. Sequential pac learning. In *Conference on Learning Theory*, 1995.

[21] R. S. Sutton and A. G. Barto. *Reinforcement Learning.* The MIT Press, 1998.

[22] P. Turney. Cost-sensitive classification: empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research*, 1995.

[23] P. Turney. Types of cost in inductive concept learning. In *Workshop on cost sensitive learning (ICML)*, 2000.

[24] G. M. Weiss and F. Provost. Learning when training data are costly: the effect of class distribution on tree induction. *Journal of Artificial Intelligence Research*, 2003.

APPENDIX

A. PROOFS

A.1 Proposition 1

We use two lemmas to aid in the proof. The first is a standard result from the theory of partitions [1]:

Lemma 1. *There are* $\binom{n-1}{b-1}$. *ways to express an integer* $n \geq 1$ *as the sum of exactly* b *positive integers.*

while the second lemma can be derived from the first [17]:

Lemma 2. *There are* $\sum_{j=1}^{b} \binom{n-1}{j-1}\binom{b}{j} = \binom{n+b-1}{b-1}$. *ways to express an integer* $n \geq 1$ *as the sum of at most* b *positive integers.*

Proof of Proposition 1. Let $d = |S| \sum_i |X_i|$. Working from the bottom-up, the dynamic program must begin by calculating the value of all possible "end states", where an end state corresponds to a complete allocation of the learning budget, b_L, over the possible purchases. Using our Naïve Bayes assumption and the unit cost of features, each purchase can result in d possible outcomes. Thus, at the bottom level, the dynamic program (dp) must consider the value of every distinct subproblem q, where q corresponds to a complete allocation of the learning budget b_L over the possible outcomes d.

Stated this way, the number of distinct subproblems q that the dynamic program has to solve is exactly the number of ways to compose the learning budget b_L of at most d positive parts.

Using Lemma 2, at the bottom level, the dp must solve

$$\frac{(b_L + d - 1)!}{(b_L)!(d-1)!} \tag{11}$$

subproblems; using Stirling's formula on each factorial, we get

$$\begin{aligned}
\frac{(b_L + d - 1)!}{(b_L)!(d-1)!} &> \\
\frac{\left(\frac{b_L+d-1}{b_L}\right)^{b_L}\left(\frac{b_L+d-1}{d-1}\right)^{d-1}\sqrt{2\pi(b_L+d-1)}}{\sqrt{2\pi b_L}\sqrt{2\pi(d-1)}(1+\frac{1}{11 b_L})(1+\frac{1}{11(d-1)})} &> \\
\frac{\left(\frac{b_L+d-1}{b_L}\right)^{b_L}\left(\frac{b_L+d-1}{d-1}\right)^{d-1}}{2\sqrt{2\pi}\sqrt{d-1}} & \\
\in \quad \Omega\left(\left(\frac{b_L+d-1}{b_L}\right)^{b_L}\left(\frac{b_L+d-1}{d-1}\right)^{d-1}(d-1)^{\frac{-1}{2}}\right) &
\end{aligned}$$

and the result follows using $t = (d-1)$. □

A.2 Proposition 2

To prove Proposition 2, the following lemma is required.

Lemma 3. *Let the Naïve Bayes assumption hold, and consider any set* D *of Dirichlets over the feature-class pairs and a bounded active classifier* BAC_D *(with bound* b_C*) constructed from* D*. Given any set of Dirichlets* D' *where* D' *can be made equal to* D *by specifying exactly one proper permutation for each feature, then there exists a bounded active classifier* $BAC_{D'}$ *(also with bound* b_C*) constructed from* D' *such that the expected error of* BAC_D *is equal to the expected error of* $BAC_{D'}$.

Proof. Let $P(.)_D$ denote a probability under D, and $P(.)_{D'}$ denote a probability under D'. Let b be a branch of BAC_D, which, without loss of generality, specifies some feature values $(X_i = x_i, X_j = x_j)$, and has classification label $Y = y$. Then the expected accuracy of branch b is

$$\begin{aligned}
P(X_i = x_i, X_j = x_j, Y = y)_D &= \\
P(X_i = x_i | Y = y)_D P(X_j = x_j | Y = y)_D P(Y = y) &= \\
P(X_i = x'_i | Y = y)_{D'} P(X_j = x'_j | Y = y)_{D'} P(Y = y) &
\end{aligned}$$

where x'_i is the image of x_i under the proper permutation for X_i. Thus we have converted a branch b of BAC_D into a new branch b', where the expected accuracy of b' under D' is the same as the expected accuracy of b under D. We can repeat this conversion for each branch of BAC_D to get a set of new branches which, when summed together, have the same expected accuracy as BAC_D. Of course, since the expected misclassification error is 1−(expected accuracy), the new branches have the same expected misclassification error as BAC_D as well.

All that remains to be shown is that the set of new branches forms a valid BAC with bound b_C. To see this, note that we can apply our transformation by doing a pre-order traversal of BAC_D, where at each non-leaf node specifying feature X_k, we reorder its subtrees using the proper permutation for feature X_k. A reordering of subtrees cannot invalidate the BAC, nor can it increase the bound b_C. Once the entire tree has been traversed, we are guaranteed to have applied our transformation to each feature of each branch, ensuring that each branch has been fully converted. The converted tree is the desired $BAC_{D'}$. □

Proof of proposition 2. Let us adopt the notation that D_A denotes the Dirichlets of state A. Further, let $D_A + (ijd)$ denote the Dirichlets of state A after observing $X_i = d$ on a $Y = j$ instance. Finally, let f_i denote the proper permutation for feature X_i, $V^{\pi^*}(p)$ denote the expected value of state p when following an optimal policy, and $dom(X_i)$ denote the domain of feature X_i.

The proof follows from induction on b_L. In the base case, $b_L = 0$. Since no learning budget remains in state A or B, there is no action to take, and hence trivially state A and B have the same (null) optimal action. When $b_L = 0$ the value of state A under an optimal policy is simply the expected misclassification error of the BAC^* constructed from state A's Dirichlets. By Lemma 3, state B must have a corresponding BAC with exactly the same expected misclassification error. Furthermore, the value of state B under an optimal policy cannot be any less, for if it were, then Lemma 3 implies that state A must have a corresponding BAC with lower expected error, which is a contradiction to the definition of BAC*. Thus states A and B have identical values under the optimal policy for the base case.

For the inductive step, assume the result holds for $b_L = n - 1$, and let states A and B have $b_L = n$. Now consider taking *any* initial action from state A, and then following an optimal policy. Without loss of generality, assume the action is to purchase $x_i^{(j)}$. Then the value of such an action is:

$$\sum_{d \in dom(X_i)} P(X_i = d | Y = j)_{D_A} V^{\pi^*}(D_A + (ijd), b_L = n-1) =$$

$$\sum_{d \in dom(X_i)} P(X_i = f_i(d) | Y = j)_{D_B} V^{\pi^*}(D_A + (ijd), b_L = n-1) =$$

$$\sum_{d \in dom(X_i)} P(X_i = f_i(d) | Y = j)_{D_B} V^{\pi^*}(D_B + (ijf_i(d)), b_L = n-1)$$

where the last equality follows by an application of the inductive hypothesis, since $D_A + (ijd)$ can be made equal to $D_B + (ijf_i(d))$ by using the r proper permutations, one for each feature. Thus, we have just shown that the value of an action in state A is equal to the value of the same action from state B, when the action is followed by an optimal policy. This implies that the value of the two states under an optimal policy is equal, and that the two states have identical optimal actions. $\square$