Appears in the
Proceedings of the Thirteenth International Conference on Machine Learning (IMLC-96),
Bari Italy, July 1996.

Learning Active Classifiers

Russell Greiner
Siemens Corporate Research
755 College Road East
Princeton, NJ 08540-6632
greiner@scr.siemens.com

Adam J. Grove
NEC Research Institute
4 Independence Way
Princeton, NJ 08540
grove@research.nj.nec.com

Dan Roth
Dept. of Appl. Math. & CS
Weizmann Institute of Science
Rehovot 76100, Israel
danr@wisdom.weizmann.ac.il

Abstract

Many classification algorithms are "passive", in that they assign a class-label to each instance based only on the description given, even if that description is incomplete. In contrast, an *active* classifier can — at some cost — obtain the values of missing attributes, before deciding upon a class label. The expected utility of using an active classifier depends on both the cost required to obtain the additional attribute values and the penalty incurred if it outputs the wrong classification. This paper considers the problem of *learning* near-optimal active classifiers, using a variant of the probably-approximately-correct (PAC) model. After defining the framework — which is perhaps the main contribution of this paper — we describe a situation where this task can be achieved efficiently, but then show that the task is often intractable.

1 INTRODUCTION

A *classifier* is a function that assigns a class label to an instance. For example, given information about a patient (such as symptoms and test values), a diagnostic classifier might specify the disease; given a visual scene of the world, a visual classifier might decide what object is being depicted; etc. Many classifiers (including most of those considered by researchers attempting to *learn* classifiers) have no control over how much data they see. A more versatile classifier, however, might first seek additional information about the instance before deciding upon a classification. As obtaining data usually involves costs — *e.g.*, to perform a medical test, or to run an specialized image processor — a classifier should not necessarily request all possible pieces of information. We therefore define an *active classifier* as a function which, given a partially specified instance, returns either a class-label or a strategy that specifies which test should be performed next.

As an example, suppose that a medical classifier is initially told only that a patient is jaundiced (*i.e.*, her eyes are yellowish). A passive classifier must then return either the diagnosis that the patient has hepatitis, or the diagnosis that she does not. An active classifier could return either of these responses, or it could perhaps order a blood test, and if that test is positive, return the diagnosis "hepatitis", but if the blood test is negative, then order a liver biopsy, and decide on the diagnosis based on the result of this test.

In the standard machine-learning paradigm, a classifier is considered good precisely if it correctly identifies the class label for as many of the instances as possible. This measure is too simplistic for active classifiers. The correct measure is decision theoretic, balancing the costs of acquiring addition information against the penalties for incorrect classification. For instance, it may not be worth spending \$1,000 to perform an expensive test to distinguish two minor variants of hepatitis, especially if the treatment is the same for both [PP91]; similarly, it is not appropriate to spend \$100 to obtain the information required to win a \$1 bet.

On any given instance, an active classifier *ac* has a *total cost*, defined as the final penalty plus all costs incurred. Ideally, we would like to find an active classifier whose *expected total cost*, over the distribution of instances that the classifier encounters, is (near) minimum.

Section 2 contrasts this problem with previous related work and Section 3 formally defines our framework. The most distinctive aspect of our proposal is that we look at the problem of learning active classifiers in an *integrated* fashion, as opposed to the "two stage" approach, of first learning the underlying concept and then, in a separate phase that does not involve learning, finding the best active classifier. Section 4 argues that this idea has the potential to improve over the two-phase approach. The rest of the paper is a (preliminary) investigation as to whether this potential can be realized; the news here is mixed. Section 5 demonstrates an interesting case in which active classifiers can be learned efficiently. But Section 6 then shows that the problem is very often intractable. The conclusion, Section 7, presents some ideas for future work, and some thoughts on the contrast between learning active versus passive classifiers.

2 LITERATURE SURVEY

Our framework is based upon the "standard" learning model [BMSJ78], in which a learner receives a set of labeled (*i.e.*, correctly classified) training examples as input, and must output a good classifier. Furthermore, the notion of "good" we use is a derivative of the popular probably-approximately-correct (PAC) model [Val84]. However, we differ from the usual model in (at least) the following respects. First, our classifier receives only *partially specified* instances, which can omit the values of some or all attributes. Note, though, that we assume the learner has access to complete instances.[1] Second, our classifier is able to *actively* request attribute values. Third, the quality of such a classifiers depends on the *expected cost of obtaining attributes*, as well as its classification accuracy.

Missing attribute values: Several other learning algorithms produce classifiers that can deal with partially specified instances; *cf.*, [DLR77, LR87, Qui89, SG94]. However, these classifiers are not able to actively obtain missing information. [BDD93, KR95] consider the problem of learning from partially specified instances, but with the goal of (resp.) later classifying *complete* instances, or later reasoning with respect to the learned concept. [GGK96] also considers learning from partially specified instances, but in situations where this missing information is known not to matter to the classification.

Active classification: "Active" classification is not a novel concept; many diverse areas use related ideas (including planning, diagnosis, decision theory, and so on). As just one illustrative example, Heckerman *et al.* [HBR94] describe how to translate any given Bayesian net (which satisfies certain properties) into (what we call) an effective "active classifier" that both isolates and repairs the fault, taking account of costs and the probability of various diagnoses being correct. However such work usually does not consider the problem of *learning* such classifiers. One possible reason is that the tasks of learning and classifying can often be decoupled from each other. For instance, [HBR94] could appeal to standard Bayesian-network learning techniques to learn the necessary distributions. While conceding that such a decoupling is possible in many cases, the basic question examined in this paper is whether there can be any advantage in studying *learning and active classification together*; see Section 4.

Our task, of learning active classifiers, should also be distinguished from another interesting problem, that of actively learning (passive) classifiers. For example, [Ang87, Ang88], [KMT93] consider models in which the learner can query for *labels* of examples as it is learning. In contrast, our *learner* is passive; see Section 7.

Utility: There are several learning projects that attempt to learn classifiers that are sensitive to test costs. For example, Turney [Tur95] (and others; see references therein) uses heuristic methods to build decision trees which minimize classification and test costs; by contrast, we are seeking provably optimal active classifiers, of any representation. Haussler [Hau92] studies a decision-theoretic generalization of the PAC model, in which the learner may output a classification or a decision rule with the goal of minimizing a given loss function. However, his classifier always receives complete instances, and so is not active in our sense.

A final contrast is with work that assumes that a learner gets to see a (hopefully good) active classifier in action, and tries to learn to duplicate its performance [Kha96]. In our model, however, the learner must use the cost structure to discover its own classification strategy.

3 FRAMEWORK

To simplify the presentation, we make the fairly standard assumption that all attributes, as well as the classification itself, are binary.[2] Thus we can identify each domain instance with a finite vector of boolean attributes $\tilde{x} = \langle x_1, \ldots, x_n \rangle$. Let $X_n = \{0,1\}^n$ be the set of all possible domain instances. We assume the world corresponds to a concept φ, which we view as an indicator function $\varphi\colon X_n \mapsto \{T, F\}$, where $\tilde{x} \in X_n$ is a member of φ iff $\varphi(\tilde{x}) = T$. We assume the learner

[1] Of course, the classifier will be tested on instances drawn from the same distribution on which the learner was trained; the examples differ only in that the classifier may initially see the values of fewer attributes. Section 3 explains why it is reasonable to give the learner this more complete information.

[2] Extensions to non-binary attributes and class-labels are straightforward. In particular, in natural problems the appropriate classification can be a member of some arbitrary (finite) set. Note that even if the concept is conceptually "binary", it is often useful to have an "I-don't-know" option available to the classifier, in addition to T and F.

knows the set of possible concepts, $\mathcal{C} = \{\varphi_i\}$ (as well as a representation scheme for these concepts).

A (labeled) example of a concept $\varphi \in \mathcal{C}$ is a pair $\langle \tilde{x}, \varphi(\tilde{x})\rangle \in X_n \times \{T, F\}$. We assume there is a stationary distribution $P : X_n \mapsto [0,1]$ over the space of domain instances, from which random instances are drawn independently, both during training and testing of the learning algorithm.

To continue the earlier example, suppose the first attribute x_1 in the instance $\tilde{x} = \langle x_1, x_2, x_3\rangle$ corresponds to the jaundice test and x_2 and x_3 correspond (respectively) to particular tests of the patient's blood and liver. Then the instance $\langle 1, 0, 1\rangle$ corresponds to a patient whose blood test was negative, but whose jaundice and liver tests (x_1 and x_3) were both positive. Assume that the concept associated with hepatitis corresponds to any tuple $\langle x_1, x_2, x_3\rangle$ where $x_1 = 1$ and either $x_2 = 1$ or $x_3 = 1$. Hence labeled examples of the concept hepatitis include $\langle\langle 1,0,1\rangle, T\rangle$, $\langle\langle 1,0,0\rangle, F\rangle$, and $\langle\langle 0,1,1\rangle, F\rangle$. Further, $P(\tilde{x})$ specifies the probability of encountering a patient with the particular set of symptoms specified by $\tilde{x}$; *e.g.*, $P(\langle 1,0,1\rangle) = 0.01$ means 1% of the time we will deal with a patient with positive jaundice and liver tests, but negative blood test.

Blocking: In the standard ("passive") model, each unlabeled instance $\tilde{x}$ is passed as is to the classifier. By contrast, this paper concentrates on the case in which the classifier initially sees no attribute values, *i.e.*, it sees $\langle *, *, \ldots, *\rangle$. We call this *complete blocking.* Of course, our classifier is subsequently allowed to obtain (at a price) the values of some blocked attributes.

A natural extension of this model considers a separate *blocking process* β which may initially present some attribute values to the classifier "for free"; *i.e.*, the classifier will initially see some element of $\{0, 1, *\}^n$. This extended model raises a number of interesting questions (*e.g.*, how does β decide which attributes to reveal? [SG94]), which space limitations prevent us from pursuing; but see the brief discussion in Section 6.

Active Classifier: An active classifier $ac : X_n^* \mapsto \{T, F\} \cup X$ is a function that maps each such partial instance $\langle x_1^*, x_2^*, \ldots, x_n^*\rangle \in \{0, 1, *\}^n$ to one of $\{T, F, 1, \ldots, n\}$, where $ac(\langle x_1^*, x_2^*, \ldots, x_n^*\rangle) = T$ (resp., F) means the classifier returns the categorical answer T (resp., categorical F). Returning $ac(\langle x_1^*, x_2^*, \ldots, x_n^*\rangle) = i \in \{1, \ldots, n\}$ means the classifier is requesting the value of the x_i attribute. Hence, $ac(\langle 1, *, *\rangle) = 2$ means the classifier is requesting the value of x_2 — *i.e.*, asking to perform the blood test on the patient. The classifier then calls itself on the result, say $\langle 1, 0, *\rangle$, perhaps to return $ac(\langle 1, 0, *\rangle) = F$. In general, of course, repeated calls to ac might be necessary (thus requesting the values for several variables) before a final answer is produced.

The learner's task may be to find the optimal active classifier amongst the set of all possible classifiers $\mathcal{A}$, or to find the best classifier of some particular type $\mathcal{A}' \subset \mathcal{A}$. In general, $\mathcal{A}$ or $\mathcal{A}'$ should be viewed as a particular "programming language" — a representation language and computational model for some (possibly restricted) class of active classifiers. Given some $\mathcal{A}'$ it then makes sense to talk of the size of an active classifier, $|ac|$, as well as its running time (*i.e.*, the time it takes ac, given input $\tilde{x}^* \in \{0, 1, *\}^n$, to output its recommendation). Naturally, we are interested in finding classifiers whose size is polynomial in the relevant quantities (such as $|\varphi|$, the size of the true concept). Furthermore, we want active classifiers that are "fast" to execute. As any particular active classifier only has finitely many inputs, we cannot speak of its asymptotic execution time in the sense of standard complexity theory. We can, however, impose the requirement of efficient execution indirectly, as a property of the computational model given by $\mathcal{A}'$.[3] In this paper we restrict attention to $\mathcal{A}'$ with the property that

> there is some fixed polynomial $p_{\mathcal{A}'}(\cdot)$ such that, for all $ac \in \mathcal{A}'$, the running time of ac is at most $p_{\mathcal{A}'}(|ac|)$.

The question of how best to represent classifiers is a subtle one, but largely beyond the scope of this paper. In the following we occasionally refer to a very simple *lookup-table* representation language. In this, one simply lists the classifier's recommendations for various tuples $\tilde{x}^* \in \{0, 1, *\}^n$; if the classifier sees a tuple that is not on the list, it performs some constant action (perhaps announce the classification F). The size of an active classifier thus represented is just the length of the given list, and the run-time complexity of using such a classifier is at most linear in its size.

Total Cost: To evaluate the quality of an active classifier, we assume as given a cost function $c_i = c(i) \in \Re$ (for $i = 1 \ldots n$) which specifies the cost of obtaining the value of the i^{th} attribute x_i; and a penalty function $\text{err}(v_1,\ v_2)$, which specifies the penalty for returning $v_1 \in \{T, F\}$ when the correct answer is $v_2 \in \{T, F\}$. Without loss of generality, we can assume $\text{err}(T,\ T) = \text{err}(F,\ F) = 0$, rescaling the other $\text{err}(\cdot, \cdot)$ values if necessary. To avoid triviality, we also assume that $\text{err}(T,\ F), \text{err}(T,\ F) > 0$.

Suppose that $\tilde{x}^* \in \{0, 1, *\}^n$ represents the classifier's current knowledge about the instance $\tilde{x} \in X_n$. We can then determine the "total cost", $tc_{ac}(\tilde{x}, \tilde{x}^*) \in \Re$, that ac would spend to complete the classification (together with the misclassification penalty, if appropriate). We

[3]This is very similar to the standard PAC requirement that the output representation can be evaluated efficiently.

define $tc_{ac}(\tilde{x}) = tc_{ac}(\tilde{x}, \langle *, *, \ldots * \rangle)$ as the total cost when the classifier begins with a completely blocked instance.

We determine $tc_{ac}(\tilde{x}, \tilde{x}^*)$ recursively. If $ac(\tilde{x}^*) \in \{T, F\}$, then $tc_{ac}(\tilde{x}, \tilde{x}^*) = \text{err}(ac(\tilde{x}^*), \varphi(\tilde{x}))$ where φ is the target formula. Otherwise, if $ac(\tilde{x}^*) = i \in \{1, \ldots, n\}$, then $tc_{ac}(\tilde{x}, \tilde{x}^*) = c(i) + tc_{ac}(\tilde{x}, \tilde{x}^*_{i \mapsto \tilde{x}[i]})$ where $\tilde{x}^*_{i \mapsto \tilde{x}[i]}$ is the result of replacing the variable indexed by $i = ac(\tilde{x}^*)$ with the value of that $\tilde{x}$ specifies for this variable, $\tilde{x}[i]$. As an example, suppose $\tilde{x} = \langle 1, 0, 1 \rangle$, $\tilde{x}^* = \langle *, *, * \rangle$ (*i.e.*, we have not yet asked for any attribute's value), and $ac(\tilde{x}^*) = 2$. Then $\tilde{x}^*_{2 \mapsto \tilde{x}[2]} = \langle *, 0, * \rangle$, as it replaces the $\tilde{x}^*_2$ value with $\tilde{x}[2] = 0$. If we suppose further that $ac(\langle *, 0, * \rangle) = F$, then

$$\begin{aligned} tc_{ac}(\langle 1,0,1 \rangle) &= tc_{ac}(\langle 1,0,1 \rangle, \langle *,*,* \rangle) \\ &= c[ac(\langle *,*,* \rangle)] + tc_{ac}(\langle 1,0,1 \rangle, \langle *,*,* \rangle_{2 \mapsto \langle 1,0,1 \rangle[2]}) \\ &= c(2) + tc_{ac}(\langle 1,0,1 \rangle, \langle *,0,* \rangle) \\ &= c_2 + \text{err}(ac(\langle *,0,* \rangle), \varphi(\langle 1,0,1 \rangle)) \\ &= c_2 + \text{err}(F, T) \end{aligned}$$

We define the "expected total cost" of the active classifier ac under the distribution of instances $\tilde{x}$, P, as $Ec_P(ac) = E_{\tilde{x} \in P}[\, tc_{ac}(\tilde{x})\,] = \sum_{\tilde{x} \in X_n} P[\tilde{x}] \times tc_{ac}(\tilde{x})$.

Performance Criterion: We assume that a learning algorithm L can draw random correctly-labeled completely-specified examples $\langle \tilde{x}, \varphi(\tilde{x}) \rangle$ according to the distribution P. (One possible justification for allowing the learner to train on *complete* instances (even though the classifier will see only *partial* instances) is that it can be cost-effective to invest in a relatively expensive training phase, if we expect that the active classifier we learn will be used very often. In this case, the cost of obtaining all attributes while learning, amortized over a much longer performance phase, might be insignificant. Although we plan to later consider training on incomplete instances, this is beyond the scope of the present work; see Section 7.)

We will evaluate the learner L in terms of the expected total cost of its output, $ac^{(L)}$. For any such $\varphi \in \mathcal{C}$ and $\mathcal{A}'$, let $ac_{\varphi, \mathcal{A}', P} \in \mathcal{A}'$ be an active classifier whose expected total cost is minimum among classifiers in $\mathcal{A}'$: $\forall ac \in \mathcal{A}'$, $Ec_P(ac_{\varphi, \mathcal{A}', P}) \le Ec_P(ac)$. (When the dependence on $\mathcal{A}'$ and P is clear, we will write ac_φ rather than $ac_{\varphi, \mathcal{A}', P}$.)

We define the following variant of the standard "Probably Approximately Correct" (PAC) criterion [Val84, KLPV87] to specify the desired performance of such a learner.

Definition 1 (PAO-AC-Learning) *Given a set of concepts $\mathcal{C}$, a class of distributions $\mathcal{D}$, a cost function $c(\cdot)$, and a penalty function $err(\cdot, \cdot)$, we say that an algorithm L* PAO-AC-learns *a set of active classifiers $\mathcal{A}'$ (w.r.t. $\mathcal{C}$, $\mathcal{D}$, $c(\cdot)$ and $err(\cdot, \cdot)$) if, for some polynomial function $p(\cdots)$, for any target concepts $\varphi \in \mathcal{C}$, distribution $P \in \mathcal{D}$, and error parameters $\epsilon, \delta > 0$, L runs in time at most $p(\frac{1}{\epsilon}, \frac{1}{\delta}, |\varphi|)$, and outputs an active classifier $L(\epsilon, \delta, \mathcal{C}, \mathcal{D}, c(\cdot), err(\cdot, \cdot)) = ac^{(L)} \in \mathcal{A}'$, whose expected total cost is, with probability at least $1 - \delta$, no more than ϵ over the minimal possible expect total cost; i.e.,*

$$\begin{aligned} &\forall P \in \mathcal{D},\ \varphi \in \mathcal{C},\ \epsilon, \delta > 0, \\ &\quad P(\ \mathrm{Ec}_P(ac^{(L)}) > \mathrm{Ec}_P(ac_{\varphi, \mathcal{A}', P}) + \epsilon\ ) \quad \le \quad \delta\ . \end{aligned}$$

Note that the number of samples drawn by $ac^{(L)}$ can be no more than the running time, and thus is polynomial in $\frac{1}{\epsilon}, \frac{1}{\delta}, |\varphi|$. Similarly, the size of the learned classifier, $|ac^{(L)}|$, is bounded by the learner's running time, and so is polynomial as well. Using the requirement that $\mathcal{A}'$ includes only classifiers whose execution time is time polynomial in their size, we see that $ac^{(L)}$'s run-time is also polynomial in $\frac{1}{\epsilon}, \frac{1}{\delta}, |\varphi|$.

Finally, while this definition allows restricting the possible distributions P to some class $\mathcal{D}$, we will focus on the normal ("distribution-free") case, where $\mathcal{D}$ includes all possible distributions.

4 *WHY* LEARN *ACTIVE CLASSIFIERS?*

The optimal active classifier is determined by the concept φ, the set of active classifiers $\mathcal{A}'$, and the distribution P. If we know all of these, then then we are faced with a very interesting optimization problem [HBR94] — one which, however, has nothing to do with learning. Sometimes this problem is tractable as, for instance, in the following case involving product distributions (*i.e.*, distributions in which each attribute's value is determined independently) and classifiers that can only ask a constant number of questions.

Proposition 1 *Suppose we know the concept φ and the* product *distribution over instances P, and are considering only the set of active classifiers that ask for at most k attribute values, $\mathcal{A}^k$. Then, for any $\epsilon > 0$, there is an efficient algorithm that runs in time $O(poly(n, \frac{1}{\epsilon}))$ and produces an active classifier whose expected total cost is within ϵ of the optimal active classifier in $\mathcal{A}^k$.*

Proof: (Sketch)[4] As P is a product distribution, the classifier can generate "simulated" data from P that matches the given instance. As it also knows φ, it can estimate the probability of T versus F and thus determine which is the better response. In general,

[4]All proofs are sketched or omitted entirely, due to space limitations. The extended report, [GGR96], containing all proofs, is in preparation.

it can also determine whether it should ask for the value of an attribute using straightforward *dynamic programming*; see Section 5 below. ▮

This suggests an obvious way to learn an active classifier: first learn the optimal underlying (passive) classifier φ and the distribution P, and then combine these to produce the best active classifier $\mathrm{ac}_{\varphi,\mathcal{A},P}$. While Proposition 1 shows that this "learn then optimize" approach can sometimes works, there are problems. First, and unsurprisingly, the optimization problem can be intractable:

Proposition 2 *There exists choices for $\mathcal{A}'$, P, $\mathcal{C}$ such that the problem of finding an approximately optimal $\mathrm{ac}^{AO} \in \mathcal{A}'$, given $\varphi \in \mathcal{C}$, with $\mathrm{Ec}_P(\mathrm{ac}^{AO}) \leq \mathrm{Ec}_P(\mathrm{ac}_\varphi) + \epsilon$, is NP-hard.*

This result holds even if we further require that $|\mathrm{ac}_\varphi|$ be polynomial in $|\varphi|$ (i.e., the complexity is not simply because a very long classifier is needed), that $\mathcal{C}$ be PAC-learnable, and that P have support of size $O(n)$.

We include the observation that the result holds even if $\mathcal{C}$ is PAC-learnable, and the distribution has "small" support (in comparison to the potential support size of 2^n, where n is the number of attributes), to emphasize that the complexity is, in a sense, "independent" of the complexity of learning. Learning the concept and/or the distribution poses separate problems:

Proposition 3 *There are some concept classes $\mathcal{C}$ (together with $\mathcal{A}'$, P, $err(\cdot,\cdot)$, $c(\cdot)$) such that finding the optimal active classifier is trivial if we are given $\varphi \in \mathcal{C}$, but otherwise is not known to be possible.*

Proof: This claim reduces to the fact that not everything is known to be PAC learnable [Ang92] because, if all costs $c(x_i)$ are zero, the classifier can ask for all attributes and then we will classify optimally if and only if it knows the concept. ▮

The preceding Propositions show that, while the "learn then optimize" approach is certainly *sufficient* (in principle) to determine ac_φ, it can fail (for complexity reasons) in various ways. *Our paper's main point, however, is that it may be easier to simply learn the active classifier directly.* In particular, one can sometimes learn a good active classifier *without* having learned (even implicitly) the concept or the distribution. This basic idea — of learning just enough to perform some particular task, rather than trying to learn everything — has been used by Khardon and Roth [KR94] in their *Learning to Reason* framework. They are concerned with the task of logical reasoning (rather than classification), and show that there can be significant complexity advantages in directly learning a representation tailored to a particular reasoning task (rather than trying to learn the concept itself and then, in a separate phase, perform logical deduction). Our work shares very much the same philosophy (if none of the technical underpinnings) as Learning to Reason.

When might it be a good idea to learn the active classifier directly? Our main positive result, given in Section 5, provides one answer in detail. Below are some of the underlying general issues:

- We do not always have to learn the full concept. The following simple example shows why. Suppose $\mathrm{err}(\cdot,\cdot)$ and $c(\cdot)$ are such that it is never worthwhile asking more than one question. Then the optimal active classifier is completely determined once we specify which attribute we should request, and which classification (T or F) is most likely given each value that this attribute might take (forming "decision-stumps" of the form studied in [Hol93, AHM95]). We can sometimes discover this classifier without knowing the full concept itself. Of course, knowing the full concept would be important if we were frequently asked to classify complete (unblocked) instances. But this is simply irrelevant: as we know that attributes will be presented completely blocked, we know that such questions will not in fact be asked. We should only care about cases that we actually might encounter (with high enough probability).

- We do not always need to learn the complete distribution P. The same example shows that, in some cases, only a few aspects of the distribution may be relevant: here we only need to know correlations between single attributes and the class label. Higher order correlations (*i.e.*, involving more than one attribute) do not affect the optimal active classifier.

- There is a second reason why we might not need to learn the distribution. The standard PAC-learning framework usually avoids having to learn distributions, because the performance criterion uses the same distribution that one learns under. If one has a (passive) classifier that fits the sample data well enough, one may hope that it will perform well on other data from the same distribution. We do not necessarily need to know *what* that distribution is; only that it has not changed since the learning phase. As our definition of PAO-AC-learning is similar to the standard PAC formulation in this respect, it too might avoid the need to learn distributions.

Of course, these arguments are only suggestive. Section 6 below will show several significant limitations on what can be achieved. However, we first present a straightforward, yet worthwhile, positive result.

5 LEARNING $\mathcal{A}^k$ UNDER COMPLETE BLOCKING

This section presents a nontrivial class of problems for which we can efficiently learn the optimal active classifiers. Our assumption of complete blocking is critical here. We also restrict the set of active classifiers considered: we only consider classifiers that request (at most) a constant, k, attribute values. (The particular attributes requested can vary from instance to instance.) To motivate seeking only such classifiers, consider a time-critical task, where a classification returned after k seconds is useless (perhaps because we know the patient will be dead by then). Similarly, a health-plan may be willing to spend up to k dollars to diagnose a specific ailment per patient, which means the doctor should not consider any classification process that can require spending more than this amount.

Formally, we are looking for a good classifier in $\mathcal{A}^k$ — the class of all active classifiers ac such that $ac(\tilde{x}^*) \in \{T, F\}$ whenever $\tilde{x}^*$ has k specified values (*i.e.*, $n-k$ $*$'s), and gives a constant response (*e.g.*, F) if there are more than k specified values. Notice that the size of such a classifier in the lookup-table representation is at most $O(2^k n^k)$; given complete blocking, this bounds the number of distinct contexts in which the learner might find itself.

Given these (restrictive, but not unrealistic) assumptions, it is possible to PAO-AC-learn *any* concept class $\mathcal{C}$ under *any* distribution. In particular, we can learn to actively classify (under these assumptions) with respect to concepts and distributions that are not learnable in the pure PAC-learning sense!

Our algorithm for this task, $L^{(k)}$, involves straightforward *dynamic programming*. In general, let $X^*_{n,m}$ be the set of all blocked $n-$tuples with exactly $n-m$ $*$'s (*i.e.*, m attributes have known values). Given our assumptions, any $\tilde{x}^* \in X^*_{n,k}$ can only appear once the classifier has already asked k questions and thus, as discussed above, we must have $ac_\varphi(\tilde{x}^*) \in \{T, F\}$. For such $\tilde{x}^*$, we can find the best classification simply by sampling. Suppose, for example, that P is the uniform distribution P_{uniform}. The proportion of training instances matching $\tilde{x}^*$ will be about $1/2^k$. Thus, in a reasonable number of samples, we can obtain a good estimate of $P^\varphi_{\tilde{x}^*} = P(\,\varphi(\tilde{x})\,|\,\tilde{x} \textit{ matches } \tilde{x}^*\,)$, and so can then determine the appropriate classification for $\tilde{x}^*$: As the difference in expected penalty between answering T as opposed to F is:

$$\begin{array}{rcl}(1-P^\varphi_{\tilde{x}^*})\text{err}(T,F) - P^\varphi_{\tilde{x}^*}\text{err}(F,T) & = & \\ \text{err}(T,F) \; - \; P^\varphi_{\tilde{x}^*}\,(\text{err}(T,F)+\text{err}(F,T)) & & \end{array} \quad (1)$$

the optimal classifier $ac^{(L)}$ should choose T iff

$$P^\varphi_{\tilde{x}^*} \quad > \quad \frac{\text{err}(T,F)}{\text{err}(T,F)+\text{err}(F,T)}.$$

What happens if our estimate of $P^\varphi_{\tilde{x}^*}$ is inexact — which it generally will be, due to statistical fluctuations? In general, the only potential problems arise if $P^\varphi_{\tilde{x}^*}$ is near the above threshold, as this could cause us to make the wrong decision. But, from Equation 1, this is precisely when it does not matter much which decision we make, because the expected costs are nearly the same. Do things change for distributions other than P_{uniform}? In this case, there may be some probabilities whose estimate is wildly inaccurate (because we see so few matching samples). But, by our PAC-like performance criterion, it does not matter much if we do badly on these extremely unlikely cases. We make these argument precise in the full paper, but this is the basic idea underlying $L^{(k)}$'s correctness: Estimated payoffs are *good enough* in this setting, and although they may lead to an classifier whose recommendations differ from the optimal classifier, this only happens when the disagreement does not affect costs by much.

Having decided what the $ac^{(L)}$ classifier should do for $\tilde{x}^* \in X^*_{n,k}$, we now show how to determine the correct actions for each $\tilde{x}^{*\prime} \in X^*_{n,k-1}$; and then use the same reduction to deal with each $\tilde{x}^{*\prime\prime} \in X^*_{n,k-2}$, and so on, until reaching $X^*_{n,0} = \{\langle *, *, *, \ldots, * \rangle\}$, thus completing the specification of the learned classifier $ac^{(L)}$.

To explain each step, suppose $ac^{(L)}$ has decided what to do for all $\tilde{x}^* \in X^*_{n,k-i}$ $(i > 0)$, and is considering some particular $\tilde{y}^* \in X^*_{n,k-(i-1)}$. Let U be the costs already incurred in reaching $\tilde{y}^*$. $ac^{(L)}$'s possible actions are to announce a classification (*i.e.*, T or F) or ask about a variable whose value is not yet known. The expected cost for each of these can be estimated using statistics gathered from the sample data. For instance, the cost of announcing T is simply:

$$Ec_P(\text{ac}^{(L)}[\tilde{y}^*]) \quad = \quad U + (1-P^\varphi_{\tilde{y}^*})\,\text{err}(T,F)$$

and the expected cost of testing attribute x_i is:

$$\begin{array}{rl}Ec_P(\text{ac}^{(L)}[\tilde{y}^*]) & = \\ U \; + & P(\tilde{x}_i = T|\tilde{x} \textit{ matches } \tilde{y}^*)\; Ec_P(\text{ac}^{(L)}[\tilde{y}^*_{i\mapsto T}]) \\ + & P(\tilde{x}_i = F|\tilde{x} \textit{ matches } \tilde{y}^*)\; Ec_P(\text{ac}^{(L)}[\tilde{y}^*_{i\mapsto F}])\end{array}$$

where, in general, $Ec_P(\text{ac}^{(L)}[\tilde{x}^*])$ is the expected cost of classifier $ac^{(L)}$ among all samples matching $\tilde{x}^*$. Note that the expected costs required by the last equation ($Ec_P(\text{ac}^{(L)}[\tilde{y}^*_{i\mapsto T}])$, $Ec_P(\text{ac}^{(L)}[\tilde{y}^*_{i\mapsto F}])$), have been estimated at the previous phase of the algorithm. Our $L^{(k)}$ algorithm then simply assigns to $ac^{(L)}$ the action with the best expected costs, based on estimates computed above.

The proof that $L^{(k)}$ correctly PAO-AC-learns a good classifier requires a sensitivity analysis, to examine how the likely error of $ac^{(L)}$ (*i.e.*, deviation from optimal) depends on the errors in statistical estimation.

Fortunately, this $ac^{(L)}$ is only likely to depart from the true optimal classifier, ac_{opt}, when the departure is of little consequence.

Theorem 4 *The learning algorithm $L^{(k)}$ PAO-AC-learns active classifiers in the set $\mathcal{A}^k$ (using the lookup-table representation, for instance), for* any *concept φ and* any *distribution P.*

6 SOME (mostly) NEGATIVE RESULTS

It seems to be surprisingly hard to relax the assumptions used in Theorem 4 while maintaining tractability. Here we investigate several reasonable weakenings.

6.1 INCOMPLETE BLOCKING

Earlier, we mentioned a generalized blocking model, in which some attributes might be revealed to the classifier "for free". To be concrete, we consider the class of *p-independent* blocking models, in which a a weight p coin is tossed (independently) for each attribute and the attribute's value is revealed iff its coin comes up heads. (Of course, the classifier still has the opportunity to ask further questions.) In analogy to the class $\mathcal{A}^k$, we define $\mathcal{A}^{+k}$ as the class of classifiers that can ask at for at most k *additional* attributes other than those the blocker reveals for free.

We distinguish three cases. First, if $p = O(1/n)$ then we expect to see only a constant number of attributes revealed. It is easy to show that a version of Theorem 4 still applies. This is simply because, with very high probability, the classifier will be given one of only polynomially many initial configurations, and we can run $L^{(k)}$ separately for each.

The $p = O(\log n/n)$ case is much more interesting, because the result is still positive even though there can be superpolynomially many starting configurations ($n^{O(\ln n)}$).

Theorem 5 *Under $O(\log n/n)$-independent blocking, there is a representation scheme under which $\mathcal{A}^{+k}$ can be PAO-AC-learned.*

Notice it is not possible to use the lookup-table representation scheme in this case. Instead, the "learning" algorithm simply records the (polynomially many) samples seen during training and gives them to the classifier which, when faced with a particular blocked instance, estimates all the necessary statistics. Thus the representation of the classifier is essentially just the sample itself — which is reminiscent of *lazy learning* [Aha96].

Finally, we consider the $p = \Omega(\mathit{constant})$ case. Unfortunately, it is fairly easy to see that this case can be extremely difficult, as it can sometimes require learning exponentially many probabilities (*e.g.*, the probabilities that $\varphi(\tilde{x})$ is T conditioned on each possible initial configuration). This problem can arise even if the concept is known and trivial.

Proposition 6 *Let $\varphi = x_1$ be the concept that is T iff x_1 is 1, and let $p \in (0,1)$ be a fixed constant. Under p-independent blocking, any representation scheme, and for any $k \geq 0$, there exists a cost structure $\langle c(\cdot), err(\cdot,\cdot)\rangle$ such that $\mathcal{A}^{+k}$ cannot be PAO-AC-learned.*

To understand this result, note that asking for x_1's value is not optimal if the values of some other attributes (which we may see for free) are correlated with x_1; making an enquiry about x_1 may not be cost-effective. (Section 7 returns to this important point.)

6.2 OTHER BLOCKERS

Since we can cope with a blocker that is expected to reveal $O(\log n)$ attributes, we might also hope to be able to solve problems in which the active classifier itself gets to choose $O(\log n)$ attributes; *i.e.*, under complete blocking, can we learn the class $\mathcal{A}^{\log n}$? This remains an open question: it is easy to see that if PAO-AC-learning $\mathcal{A}^{\log n}$ is possible, then $\log n$-depth decision trees would be PAC-learnable in the standard (passive learning) model. But even the simpler problem, of learning boolean functions that depend on only $\log n$ variables, even under the uniform distribution, is regarded as a challenging open problem [Blu94].

On the other hand, the news is not all bad here. The difficulty here concerns *computational* complexity, and not sample complexity nor the nonexistence of a good small classifier.

Proposition 7 *It is possible to learn $\mathcal{A}^{\log n}$ in the sense of Definition 1, still using only polynomially many samples and producing a polynomial-size table-lookup classifier, except that the learner may not run in polynomial time.*

That is, the learner uses a reasonable number of samples and (eventually!) outputs a small (hence efficient) active classifier. This can be useful; if the performance phase is much longer than the training phase, it may well be worth spending whatever time is necessary to find a good classifier.

6.3 "HIGH COSTS"

It might seem reasonable to suppose that if costs are "high", relative to the potential penalties, then one should only plan to ask a few questions.

Definition 8 *Without loss of generality, assume $c_1 \leq \ldots \leq c_n$; i.e., attributes are sorted by increasing cost. We then define:*

$$k(err, c) \; = \; \text{largest } k' \text{ such that} \left\{ \sum_{i=1}^{k'} c_i \; \leq \; \max_{v_1,v_2} \{ \; err(v_1, v_2) \; \} \right\}.$$

It is easy to see that:

Proposition 9 *The optimal active classifier should not ask* on average *more than $k(err, c)$ questions (assuming complete blocking).*

Unfortunately, this does not mean that we can *bound* the number of questions by $k(\mathrm{err}, c)$ — *i.e.*, we cannot restrict ourself to $\mathcal{A}^{k(\mathrm{err},c)}$.[5] The problem is that a classifier may reach a point where it should ask yet more questions, even after it has spent more than any possible payoff. This is because earlier costs are *sunk costs* and even if, in retrospect, they turn out not to have been useful, they must still be paid for. However the optimal classifier should not expect *a priori* to get into this situation very often, as a classifier that often throws good money after bad cannot be optimal.

While it may seem plausible that a variant of the dynamic programming can still PAO-AC-learn active classifiers under the assumption of constant $k(\mathrm{err}, c)$, this is wrong:

Theorem 10 *Let $\mathcal{A}^{\approx k}$ be the class of active classifiers whose members each ask, on average, at most k questions. Unless $P = NP$, no efficient algorithm can PAO-AC-learn $\mathcal{A}^{\approx 2}$.*

(Note that $\mathcal{A}^{\approx 1}$ is the same as $\mathcal{A}^1$ and so it is learnable.) This hardness result only talks about computational complexity. It is open as to whether this is the only difficulty (*i.e.*, is there a result analogous to Proposition 7?).

6.4 RESTRICTED DISTRIBUTIONS

We have seen it can be hard to learn good active classifiers even if the underlying concept is very simple; in some cases, this is due to the complexity of the distribution P. This suggests considering restricted classes of distributions. The obvious candidate is the class of "product distributions", in which each attribute value is chosen independently; the uniform distribution is a further restriction of this class. We have only just begun exploring this promising direction. On the one hand, not all difficulties are due to complex distributions: We noted above that PAO-AC-learning $\mathcal{A}^{\log n}$ subsumes a hard open problem, even under the uniform distribution. Sometimes, however, distributional restrictions can be exploited. For instance,

[5] On the other hand, it *is* easy to see that it is sufficient to consider classifiers in $\mathcal{A}^{k(\mathrm{err},c)/\epsilon}$. But then our dynamic-programming algorithm will be exponential in $1/\epsilon$.

Remark 11 *If $\mathcal{C}$ is the class of conjunctions and the distribution is known to be a product distribution, then the optimal active classifier (under any blocking model, and with any cost structure) uses a straightforward greedy strategy.*

(Very roughly speaking, this classifier always asks for the attribute that promises the highest immediate information gain about the classification, balanced by cost.)

Although this is a simple observation, it does show that the negative result in Proposition 6 depends crucially on having "difficult" distributions. Even under product distributions, greedy active classification is not guaranteed to work in general (*i.e.*, beyond the class of conjunctions); counterexamples are easy to find. However, variants of the greedy strategy might be very useful heuristics and this, too, is worth further investigation.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed a framework for studying *learning* and *active classification* together. It seems plausible that this might entail some "Learning-to-Reason"-style advantages, in that learning a particular classification strategy (w.r.t. a particular cost structure and blocker) might be easier than learning the full concept (and perhaps distribution). We have presented results which supports this thesis, but also shown that the scope for such advantages is quite restrictive. Three research directions that offer hope for further positive results are (1) other restrictions on the type of active classifiers allowed; (2) approximation techniques; and (3) the combination of stronger restrictions on both the concept class and the distribution. We are also exploring an "on-line" version of this framework. Here the learner would incur costs even while it was learning: it would pay for any attribute it sees, and would have to predict each instance's classification (risking penalty). The goal would be to minimize total cost over some lifetime. Among the new difficulties this would introduce is the characteristic dilemma of *bandit* problems [BF85]: finding the right balance between learning and performance in terms of the number of attributes requested. The model in this paper was (at the expense of some realism) designed so that this particular tradeoff is avoided.

We close by noting an interesting contrast between our results and standard PAC concept learning: *Few of our results depend, in any critical way, on the identity of* *concept class.* Theorem 4, while rather restrictive in many other respects, works for any concept at all. To explain this difference, first note that when one does not see all the attributes then the induced probabilistic concept [KS90] over the visible attributes can, in general, be quite complex, even if the real concept is a simple one. A second issue is that the distribution appears to matter more. For example, attributes which are logically irrelevant to the concept might nevertheless be very important to an active classifier, if their values are somehow correlated with the correct label. Both of these reasons suggest that, to whatever extent that active classifiers can be learned at all, we might expect to find results that do not distinguish between concept classes to the extent that ordinary passive classifier learning theory does.

Acknowledgements

We gratefully acknowledge receiving helpful comments from Mukesh Dalal, Sanjeev Kulkarni, Nick Littlestone, and Dale Schuurmans.

References

[Aha96] D. Aha. Special issue on "Lazy Learning". *Artificial Intelligence Review*, 1996. (in progress).

[AHM95] P. Auer, R. C. Holte, and W. Maass. Theory and applications of agnostic PAC-learning with small decision trees. In *ICML-95*, pages 21–29, 1995.

[Ang87] D. Angluin. Learning regular sets from queries and counterexamples. *Inform. Comput.*, 75(2):87–106, 1987.

[Ang88] D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, April 1988.

[Ang92] D. Angluin. Computational learning theory: survey and selected bibliography. In *STOC-92*, pages 351–369, 1992.

[BDD93] S. Ben-David and E. Dichterman. Learning with restricted focus of attention. In *COLT93*, pages 287–296, 1993.

[BF85] D. Berry and B. Fristedt. *Bandit Problems: Sequential Allocation of Experiments.* Chapman and Hall, London, 1985.

[Blu94] A. Blum. Relevant examples and relevant features: Thoughts from computational learning theory. In *AAAI Fall Symposium on 'Relevance'*, 1994.

[BMSJ78] B. G. Buchanan, T. M. Mitchell, R. G. Smith, and C. R. Johnson, Jr. Models of learning systems. In *Encyclopedia of Computer Science and Technology*, volume 11. Dekker, 1978.

[DLR77] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statistics Society, B*, 39:1–38, 1977.

[GGK96] R. Greiner, A. Grove, and A. Kogan. Exploiting the omission of irrelevant data. In *ICML-96*, 1996.

[GGR96] R. Greiner, A. Grove, and D. Roth. Learning active classifiers. Technical report, Siemens Corp. Res., 1996.

[Hau92] David Haussler. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Information and Computation*, 100(1):78–150, 1992.

[HBR94] D. Heckerman, J. Breese, and K. Rommelse. Troubleshooting under uncertainty. In *International Workshop on Principles of Diagnosis*, 1994.

[Hol93] R. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–91, 1993.

[Kha96] R. Khardon. Learning to act. In *AAAI-96*, August 1996.

[KLPV87] M. Kearns, M. Li, L. Pitt, and L. G. Valiant. On the learnability of boolean formulae. In *STOC-87*, pages 285–295, 1987.

[KMT93] S. R. Kulkarni, S. K. Mitter, and J. N. Tsitsiklis. Active learning using arbitrary binary valued queries. *Machine Learning*, 11(1), 1993.

[KR94] R. Khardon and D. Roth. Learning to reason. In *AAAI-94*, pages 682–687, 1994.

[KR95] R. Khardon and D. Roth. Learning to reason with a restricted view. In *COLT-95*, pages 301–310, 1995.

[KS90] M. Kearns and R. Shapire. Efficient distribution-free learning of probabilistic concepts. In *FOCS-90*, 1990.

[LR87] J. Little and D. Rubin. *Statistical Analysis with Missing Data.* Wiley, New York, 1987.

[PP91] Gregory Provan and David Poole. The utility of consistency-based diagnostic techniques. In *KR-91*, pages 461–72, 1991.

[Qui89] J. R. Quinlan. Unknown attribute values in induction. In *ICML-89*, pages 164–168, 1989.

[SG94] D. Schuurmans and R. Greiner. Learning default concepts. In *CSCSI-94*, pages 519–523, 1994.

[Tur95] P. D. Turney. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of AI Research*, 2:369–409, 1995.

[Val84] L. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–42, 1984.