Learning Policies for Sequential Time and Cost Sensitive Classification

Andrew Arnt
arnt@cs.umass.edu

Shlomo Zilberstein
shlomo@cs.umass.edu

Department of Computer Science
University of Massachusetts, Amherst
Amherst, MA 01003

ABSTRACT

In time and cost sensitive classification, the utility of labeling an instance depends not only on the correctness of the labeling, but also the amount of time taken to label the instance. Instance attributes are initially unknown, and may take significant time to measure. This results in a difficult problem, trying to manage the tradeoff between time and accuracy. The problem is further complicated when we consider a sequence of time-sensitive classification instances, where time spent measuring attributes in one instance can adversely affect the costs of future instances. We solve these problems using a decision theoretic approach. The problem is modeled as an MDP with a potentially very large state space. We discuss how to intelligently discretize time and approximate the effects of measurement actions in the current instance given all waiting instances. The results offer an effective approach to attribute measurement and classification for a variety of time sensitive applications.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning - Induction; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search - Graph and tree search strategies; H.2.8 [**Database Management**]: Applications - Data Mining

General Terms

Algorithms

Keywords

cost-sensitive learning, data mining, AO* search

1. INTRODUCTION

Cost sensitive classification (CSC) has been the subject of a growing body of research. In CSC, the goal is to train a classifier that minimizes the expected cost incurred on future test instances, rather than trying to maximize the predictive accuracy. Penalties for misclassifying instances vary based on the actual label of the instance. For example, in medical diagnosis domains classifying a sick patient as well is often far more costly than labeling a healthy patient as sick. In a spam filtering system, legitimate email flagged as spam is significantly more costly than spam judged as legitimate.

Additionally, in some CSC problems, attributes of an instance are not initially known. Instead, the CSC classifier must explicitly decide which attributes to measure. Some of these attributes may have a fixed cost to measure; an example from the medical diagnosis domain are those that require an expensive test to be performed. In this problem, an attribute measurement and classification policy that specifies what attributes to measure and in what order is designed to minimize not only misclassification penalties, but also the sum of attribute measurement costs.

In this work, we examine a previously unexplored dimension of CSC. In many domains, the value of a classification result depends not only on the correctness of the labeling, but also the timeliness with which it is computed. Furthermore, measuring some of these attributes may be either computationally intensive or rely on slow external sources of information. For example, in medical diagnosis, tests are often sent away for processing while the patients condition may be deteriorating. In the spam filtering case, retrieving or verifying hyperlinked information can take significant time and delay the arrival of email to a user's inbox. It is impractical to measure all possible attributes for each instance when the final result has time-dependent utility. We call this problem time and cost sensitive classification (TCSC).

Managing the tradeoff between classifier accuracy and time costs incurred is a challenging problem. Myopic methods such as those used in [12] will not perform well due to interactions between attributes: when not all attributes can be measured, the ordering of measurements becomes very important. There have been several methods designed by researchers for handling the CSC problem, but very little attention has been paid to the TCSC case. We develop a model that allows the system to quickly decide which attributes to measure, what order to measure them in, and when to cease any further measurement and classify the current instance.

We take a decision theoretic approach, where we try to minimize the expected value of a cost function reflecting the quality of service of the system. In the cost sensitive classifier developed in [17, 16], the attribute measurement problem was framed as a Markov Decision Process (MDP)

UBDM '05 August 21, 2005, Chicago, Illinois, USA

where the state was the current attribute vector. We build upon that work by adding the current time to that state. Due to the potentially large size of this state space, AO* heuristic search is used to compute the policy. It is not necessary in AO* to compute values for all possible states as would be required in a dynamic programming approach. The addition of time to that state space requires that time be intelligently discretized to provide a balance between the quality of the computed policy and the memory required to compute it.

We then examine the case where a sequence of time dependent instances must be classified over time. In this sequential TCSC problem, classification instances arrive at the classifier over time and are processed in a first-in first-out manner. Time spent measuring attributes in the instance at the head of the queue can increase the costs incurred on waiting instances by delaying the start of their processing. Clear examples of this type of problem are spam filtering on an overloaded mail server or estimating the value of messages posted to a newsgroup or online forum [1]. This model can also apply to diagnosis tasks. We show how to extend the MDP model to find policies that minimize cost over all instances processed and discuss the approximations used to solve this even larger MDP.

2. TYPES OF COST IN TCSC

2.1 Misclassification costs

In many applications, not all misclassifications have the same value. There may be a significant difference between the problems caused by a false negative versus those caused by a false positive. We denote this portion of the cost function which handles misclassification penalties as $C_L(l_p|l_a)$, which is the cost incurred by classifying an instance with actual label l_a with the predicted label l_p.

The misclassification (MC) cost component depends on the actual label l_a of an instance, which is unknown, except in training data. Thus, in practice we need to use the *expected* MC cost, given that the classifier predicts label l_p:

$$EC_L(\mathrm{cl}(l_p)|\mathbf{f}) = \sum_{l_a \in L} p(l_a|\mathbf{f}) C_L(l_p|l_a)$$

where L is the complete set of labels that an instance may have. The probability that an instance with the current measured attribute vector $\mathbf{f}$ has the actual label of l_a is estimated, when necessary, from training data: $p(l_a|\mathbf{f}) = \frac{|train(l_a,\mathbf{f})|}{|train(\mathbf{f})|}$, where $train(\mathbf{f})$ is the set of all training instances such that for every measured attribute in $\mathbf{f}$, the training instance has the same value, and $train(l_a, \mathbf{f})$ is the subset of instances in $train(\mathbf{f})$ that have label l_a. In practice this estimate is smoothed to reduce overfitting and avoid divisions by zero:

$$p(l_a|\mathbf{f}) = \frac{|train(l_a,\mathbf{f})| + 1}{|train(\mathbf{f})| + |L|}$$

When classifying a partially measured instance $\mathbf{f}$, the algorithm will choose the label that incurs the minimum expected MC cost on the given set of training data:

$$l_p = \arg\min_{l \in L} EC_L(\mathrm{cl}(l)|\mathbf{f}) \quad (1)$$

There has been a significant volume of work on the problem of minimizing MC costs: some general methods are the weighted boosting algorithm of [8], and the MetaCost algorithm of [7].

2.2 Attribute measurement costs

The action of measuring an attribute f_i is indicated as $\mathrm{m}(f_i)$. This action may incur a deterministic cost: $C_M(\mathrm{m}(f_i))$. We assume the value of a measured attribute is constant and will not change upon repeated measurements.

Research that handles both attribute measurement costs and misclassification costs includes the genetic algorithm based decision tree inducer of [15], the POMDP (Partially Observable MDP)-based decision tree learner of [5], a dynamic programming algorithm described in [10], a POMDP for computing attribute measurement policies with respect to a given naive Bayes classifier in [11], the test-cost sensitive naive Bayes classifer of [6], and finally an MDP framework with heuristic search to find good attribute measurement and classification policies [17, 16]. Note that none of these can handle any kind of time-dependent costs.

2.3 Response time costs

The cost function should reflect the timeliness with which we wish the classifier to act. In many systems (especially those that interact with humans), a labeling decision made quickly will be worth more than one that takes a very long time. In general, any system that has a component of human interaction should be fairly responsive and not spend unreasonable amounts of time measuring all instance attributes so as to minimize the expected MC cost.

Therefore the cost function has a final component $C_T(t)$, which will typically have a super-linear form: the cost of a quick result is small and fairly constant, but as the waiting time increases, the time cost grows at an increasing rate. This function provides a good approximation of a user's perceived utility of a system when they are forced to wait for a result. In general, the time cost function can take on any form that is nondecreasing over time.

Note that if the time cost function is linear and the measurement times for each attribute are deterministic, the time cost function can be simply folded into the attribute measurement costs.

2.4 Combining cost function components

To combine the three components of the cost function, it suffices to perform a simple weighted addition. The expected cost of assigning predicted label l_p to an instance $\mathbf{f}$ with measured attributes $\mathrm{meas}(\mathbf{f})$ in t time units is:

$$C(\mathbf{f}, t) = w_L EC_L(\mathrm{cl}(l_p)|\mathbf{f}) + w_T C_T(t) + w_M \sum_{f_i \in \mathrm{meas}(\mathbf{f})} C_M(\mathrm{m}(f_i))$$

The variables w_L, w_T, w_M are system parameters that are manually tuned to provide a good balance between the conflicting goals of low MC costs, attribute measurement costs, and timely responses.

3. SINGLE INSTANCE TCSC

Given a set of training data, we want to find the attribute measurement and classification policy that minimizes the expected cost of classification of future instances where cost is made up of the three components discussed in Section 2.

Our strategy for time and cost sensitive policy learning builds on the work of [17]. We frame the attribute measurement and classification problem as a Markov Decision

Process (MDP). The "optimal" policy (quoted because it is optimal only with respect to a set of labeled training data) can then be found using AO* search, a classical heuristic search technique. We extend this model to handle time-sensitive utility costs.

3.1 TCSC as an MDP

MDPs are a popular framework for sequential decision making problems. An agent in an MDP takes *actions* which cause stochastic transitions between *states*. A typical formulation (and the one used here) has an agent with the goal of minimizing the costs incurred while transitioning to some terminal state. Each state in the MDP satisfies the Markov property: the current state effectively summarizes all previous activity of the agent in the environment. The mapping from states to actions that minimizes cost is called the *optimal policy*.

The states $s \in S$ in the model presented in [17] are simply the set of all possible attribute vectors $\mathbf{f}$. This includes those with unmeasured attributes. An additional absorbing terminal state E is transitioned to when an instance is classified. We augment that state space to include the current waiting time of the instance: $s = \langle \mathbf{f}, t \rangle$. The starting state of the MDP is the state with no measured attributes and zero waiting time: $\langle \mathbf{f}_?, 0 \rangle$.

The actions in this model are to either measure an unmeasured attribute f_i, denoted '$\mathrm{m}(f_i)$', or to classify the current instance using the label l_p, denoted '$\mathrm{cl}(l_p)$'.

There are two types of cost related to taking a measurement action. $C_M(\mathrm{m}(f_i))$ is the deterministic cost to measure attribute f_i. There is also the incremental time cost $C_\Delta(\delta|t)$ which indicates the portion of the end cost $C_T(t)$ incurred by waiting δ additional time units to classify an instance that has already been waiting t time units. Given a time cost function $C_T(t)$, it is straightforward to compute the incremental time cost function:

$$C_\Delta(\delta|t) = C_T(t+\delta) - C_T(t)$$

The expected immediate cost of taking the action $\mathrm{m}(f_i)$ is then

$$C_M(\mathrm{m}(f_i)) + \sum_{\delta \in T_i} p(T_i = \delta) C_\Delta(\delta|t)$$

where T_i is the set of all possible durations that the measurement action can take.

The probability of transitioning from state $s = \langle \mathbf{f}, t \rangle$ to state $s' = \langle \mathbf{f} \cup f_i = x, t+\delta \rangle$ (where $\mathbf{f} \cup f_i = x$ refers to $\mathbf{f}$ with attribute f_i set to x) is

$$p(s'|s, \mathrm{m}(f_i)) = p(f_i = x|\mathbf{f}) p(T_i = \delta)$$

The probability that attribute f_i will take on value x given the incomplete attribute vector $\mathbf{f}$ is estimated from training data:

$$p(f_i = x|\mathbf{f}) = \frac{|train(\mathbf{f} \cup f_i = x)|}{|train(\mathbf{f})|}$$

In practice this value is smoothed to:

$$p(f_i = x|\mathbf{f}) = \frac{|train(\mathbf{f} \cup f_i = x)| + 1}{|train(\mathbf{f})| + |f_i|}$$

where $|f_i|$ indicates the total number of distinct values the ith attribute can have.

The probability that attribute f_i takes δ time units to measure is denoted as $p(T_i = \delta)$, and is estimated from training data or from some other source of prior experience. Note that these transition probabilities are computed only when necessary during the AO* search.

Taking the classification action incurs the MC cost $C_L(l_p|l_a)$ and transitions to the terminal state with probability

$$p(E|s, \mathrm{cl}(l_p)) = 1$$

Recall that l_p is chosen to minimize expected cost as in Equation 1.

3.2 AO* search

AO* search is an heuristic search algorithm for searching AND/OR graphs [13]. It is akin to A* search for standard directed graphs. MDP policies can be represented as an AND/OR graph: at an OR node, the agent must choose a single action to take so as to minimize future cost. However, since the environment is stochastic, taking an action causes the agent to transition probabilistically to one of a number of states. Therefore all these states are successors of the original state and their costs must be AND-ed together (computing the expected cost) to find the best expected action.

AO* works by iteratively improving upon the current best partial solution policy until an optimal policy is found. Each iteration of AO* search is composed of two parts. First, the current best partial solution is expanded (its successors are added to the search graph) by picking an unexpanded search state within the current policy. Next, state values and best action choices are updated in a bottom-up manner, starting from the newly expanded state. The estimated value of a state s during the search is $F(s)$: an optimistic estimate of the cost to get from s to a terminal state.

A heuristic is necessary to guide AO*. The heuristic value of a state is the optimistic estimate of how much cost will be incurred before reaching a terminal state. For an optimal policy to be found, the heuristic must be *admissible*: it must never overestimate the cost from a state to the terminal state. An optimistic one-step lookahead heuristic derived by [17] for the case where there is no time dependent utility was extended to include incremental time costs in [2].

Given an unexpanded state s, the heuristic value $F(s)$ is the cost of the action (classifying or measuring an attribute) giving the smallest immediate cost:

$$F(s) = \min_{f_i \notin \mathrm{meas}(\mathbf{f})} \begin{cases} EC_L(\mathrm{cl}(l_p)|\mathbf{f}) \\ C_M(\mathrm{m}(f_i)) + \sum_{\delta \in T_i} p(T_i = \delta) C_\Delta(\delta|t) \end{cases} \quad (2)$$

This heuristic is admissible because it gives the smallest possible immediate cost incurred when taking any action from the current state.

3.3 AO* as an anytime algorithm

For classification problems where instances have a large number of measurable attributes, each of which can take on many values, pruning the search space is essential for efficient search. A pruning strategy that preserves the optimality of the policy hinges on the fact that the terminal state E can be reached from any state of this MDP by taking a single classification action [17]. This property, which is not applicable for general MDP models, allows for significant pruning of the search space. An upper bound $\hat{F}(s)$ value is computed at each node; this value represents the expected

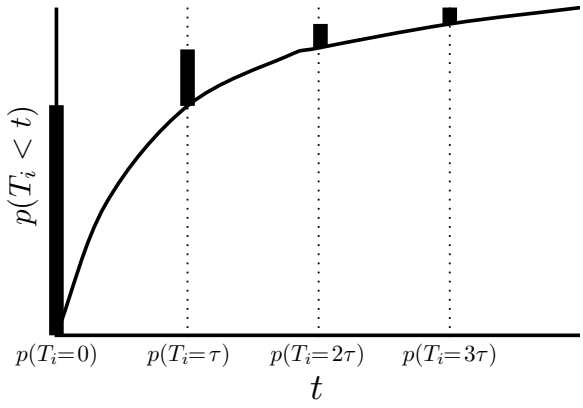

Figure 1: An attribute time distribution discretized in a 'round-down' manner. The curve shows the cumulative density function, the length of the black bars represents the probability of each discrete value.

cost of following the current *best known* policy from search state s. Therefore, any unexpanded search node s' with parent node s where $\hat{F}(s) < F(s')$ can be pruned, as the expansion of s' cannot lead to an improved policy since we will always choose the action at s that provides the minimal $\hat{F}(s)$.

Furthermore, maintaining the best known action at each search node gives the AO* search the properties of an anytime algorithm. At any time during the search, the process can be halted and the current best policy returned. For large classification problems where there are many attributes taking on many possible values in each instance, this must be done when memory is exhausted.

3.4 Discretization of time

While continuous attributes can be discretized using metrics such as information gain relative to the class attribute (for example [9]), finding an appropriate discretization of time for a instance is a more difficult problem. A very fine grained discretization of time results in the best policies, except when search terminates early due to available memory being exhausted by the larger state spaces. Furthermore, in some cases, using a coarser time representation may still find a policy of approximately equal value.

In this work we take an iterative approach to finding a suitable time unit size. Starting from an initial coarse time unit τ, we iteratively solve the MDP using a unit of $\tau' = \tau/2$. This process repeats until the cost incurred on training data by the policy $\pi_{\tau'}$ is greater than or approximately equal to the cost incurred by following π_τ; the π_τ policy is retained for actual use. Note that the cost of $\pi_{\tau'}$ can be greater than that of π_τ when the memory limit is reached and search is forced to terminate with the current best known policy.

To compute the policy for time unit τ, we use 'rounded-down' versions of the continuous attribute measurement time distributions $p(T_i)$. That is: $p_\tau(T_i = k\tau) = \int_{k\tau}^{(k+1)\tau} p(T_i = x)dx$. See Figure 1 for an example distribution. This rounding down combined with the nondecreasing nature of the time cost function $C_T(t)$ means that the F value of a state s in π_τ always underestimates of the actual cost of state s.

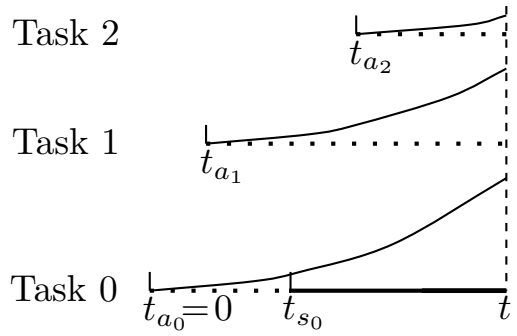

Figure 2: instances arriving over time. Dotted lines represent waiting time, solid lines active processing of an instance. Time cost curves are shown for each instance. Instance 0 arrived at t_{a_0}, but processing did not begin until t_{s_0} due to delay from measuring attributes in previous instances.

Therefore, we can use the F values computed in the τ iteration as part of a new admissible heuristic function F' in the τ' iteration; The new heuristic value at a state is the smaller of the heuristic value computed as before in Equation 2 and the final F value computed in the previous iteration (rounding down time from $k\tau'$ to $j\tau$, such that $j\tau \leq k\tau' < (j{+}1)\tau$):

$$F'(\langle \mathbf{f}, k\tau' \rangle) = \min(F(\langle \mathbf{f}, k\tau' \rangle), F_\tau(\langle \mathbf{f}, j\tau \rangle))$$

4. SEQUENTIAL TCSC

The above procedures do not account for other instances that need to be classified. Suppose that instead of a single classification instance to process, the system has to handle a *stream* of classification instances arriving over time. Therefore, when deciding which attributes to measure in the current instance, we must also consider the potential for utility loss due to delay in processing of all other instances waiting to be classified. [3] refer to this as the *opportunity cost*, the loss of expected value due to delay in the starting of work on the remaining instances. They show that for a similar problem, the opportunity cost function can be quickly and effectively approximated by examining simple attributes of the queue of waiting instances.

There are no known existing methods for classifying sequences of time sensitive instances. [14] study a sequential CSC problem where the cost of each instance is dependent on the labels assigned to prior instances. Reinforcement learning is used to minimize costs over a sequence of interacting instances; however, there is no time-sensitive cost component.

4.1 MDP model for sequential classification

We call the instance that is currently having attributes measured the 'active' instance. Time t is measured relative to the arrival time of the active instance $t_{a_0} \equiv 0$. As attributes are measured in the active instance, new instances arrive at t_{a_i}. Figure 2 shows a possible configuration.

Sequential classification instances can be introduced to the MDP model by expanding the state space to $s = \langle \mathbf{f}, t, \mathbf{q} \rangle$, where $\mathbf{q} = \{t_{a_1}, \ldots, t_{a_{|\mathbf{q}|}}\}$ describes the queue of instances waiting to be classified.

The MDP transition model must also be augmented to

include $\mathbf{q}$. The probability of transitioning from state $s = \langle \mathbf{f}, t, \mathbf{q} \rangle$ to state $s' = \langle \mathbf{f} \cup f_i = x, t + \delta, \mathbf{q}' \rangle$is

$$p(s'|s, \mathrm{m}(f_i)) = p(f_i = x|\mathbf{f})p(T_i = \delta)p(\mathbf{q}'|\mathbf{q}, \delta)$$

where $p(\mathbf{q}'|\mathbf{q}, \delta)$ is the probability of the queue going from state $\mathbf{q}$ to $\mathbf{q}'$during a time interval of δ time units. This quantity can be estimated from past experience.

We could then change the transition model so that classification actions transfer to terminal state E only when the queue is empty. In all other cases, the next state would be

$$s' = \langle \mathbf{f}_?, t - t_{a_1}, \{\forall_{1<i<|\mathbf{q}|} t'_{a_{i-1}} = (t_{a_i} - t_{a_1})\} \rangle$$

and processing begins on the new active instance. Solving this MDP would give an optimal solution to the TCSC problem. The size (possibly infinite) of this MDP makes it intractable to solve exactly.

An alternative approach is to estimate the opportunity cost of investing δ time units on the active instance given that $\mathbf{q}$ instances are waiting. Once estimated, this cost augments the incremental time cost component $C_\Delta(\delta|t)$:

$$C_{\Delta'}(\delta|t, \mathbf{q}) = C_\Delta(\delta|t) + C_{OC}(\delta|\mathbf{q}, t)$$

The MDP can be solved with the updated incremental time cost function using the techniques discussed in Section 3.

Note that instances under the model may now have start time $t \neq 0$. Therefore, when solving the MDP, a new start state is introduced where t is unknown. The only action available at this state is a 'fan-out' action which transitions to 'sub start' states with $t = 0$ to $t = T$ with uniform probability. T is chosen to be large enough so that no time consuming attributes are measured from that state. After the policy has been computed and instances are being classified online, we start the measurement policy at the appropriate sub start state. If the instance has been waiting longer than T, the policy starting at sub start state T is followed.

4.2 Estimating opportunity costs

We will examine three methods for estimating the opportunity cost incurred by delaying processing on the instances in the queue.

A very conservative estimate of opportunity cost simply sums the incremental time cost incurred on each instance in the queue, assuming that processing will begin on every one of these instances, simultaneously, after δ time units have elapsed:

$$C_{OC}(\delta|\mathbf{q}, t) = \sum_{i \in \mathbf{q}} C_\Delta(\delta|t - t_{a_i}) \tag{3}$$

A second estimate takes into account that before processing can begin on instance $i+1$, instance i must first be classified. Given a classification/measurement policy π computed on the single instance problem, the estimated time to complete the active instance given the current state can be computed in a bottom up manner. The time remaining distribution from state s given the time remaining distributions of all child states s' reached by following the action $\pi(s)$ is

$$p(t_d|s) = \sum_{s'} p(s'|s, \pi(s))p(t_d - t(s, s')|s')$$

where $t(s, s')$ is the time difference between states s and s'. With this probability, the starting times of all instances in the queue can be estimated:

$$p(t_{s_{i+1}} = t_{d_i} + t_{s_i}) = p(t_{s_i})p(t_{d_i}|\langle \mathbf{f}_?, t_{s_i} - t_{a_i} \rangle) \tag{4}$$

The start time of instance 1 is set to be the current time t of the active instance.

Once the start time distributions of all instances in the queue are computed, the opportunity cost can be estimated as the total estimated time cost incurred by delaying the estimated start times of all instances in the queue by δ:

$$C_{OC}(\delta|\mathbf{q}, t) = \sum_{i \in \mathbf{q}} \sum_{t_{s_i}} p(t_{s_i}) C_\Delta(\delta|t_{s_i} - t_{a_i}) \tag{5}$$

The above methods assume that the only costs incurred by delaying instances will be time costs. In reality, the policy for an instance that begins processing with substantial waiting time already elapsed will generally measure less time consuming attributes to avoid the continually increasing time penalties. We can then expect smaller attribute measurement costs at the expense of higher MC costs.

A third opportunity cost estimation looks at the difference in expected costs incurred for all queued instances before and after an action taking δ time units is taken in the active instance. Given the single instance policy π, the expected total cost incurred C from state s can be computed in a similar manner as the time remaining distributions:

$$p(C|s) = \sum_{s'} p(s'|s, \pi(s))p(C - c(s, s')|s')$$

where $c(s, s')$ is the cost incurred between states s and s'. With this expected cost distribution, the expected cost incurred on all queue instances can be computed, given the current time of the active instance.

$$C_{\langle \mathbf{f}, t \rangle} = \sum_{i \in \mathbf{q}} \sum_{C} \sum_{t_{s_i}} p(t_{s_i}) p(C|\langle \mathbf{f}_?, t_{s_i} - t_{a_i} \rangle) C$$

where the start time distributions $p(t_{s_i})$ are computed using equation 4 starting from time t.

The final opportunity cost estimate is the difference in expected costs when making a transition from state $\langle \mathbf{f}, t \rangle$ to $\langle \mathbf{f}', t' \rangle$ is:

$$C_{OC}(\delta|\mathbf{q}, t) = C_{\langle \mathbf{f}', t' \rangle} - C_{\langle \mathbf{f}, t \rangle} \tag{6}$$

4.3 Queue approximations

An exact representation of the state of the instance queue would contain the arrival times (relative to the arrival time of the active instance) for every waiting instance. Clearly this representation would cause an exponential blowup in the total size of the state space. Instead we choose to represent the queue using simple features describing the state of the queue. In this work, we use two features: the total number of instances in the queue and the average arrival time of those instances. A manually tuned parameter for each feature controls the resolution: as the resolution increases there are less total queue states, which results in smaller searches but lower quality opportunity cost estimates.

5. EXPERIMENTAL

5.1 Setup

We use three data sets in the following experiments: 'breast,' 'pima,' and 'bupa' from the UCI repository [4]. All attributes are discretized into three bins so as to maximize the information gain of the class labels for the entire data set. The class labels for all data sets are binary. The breast

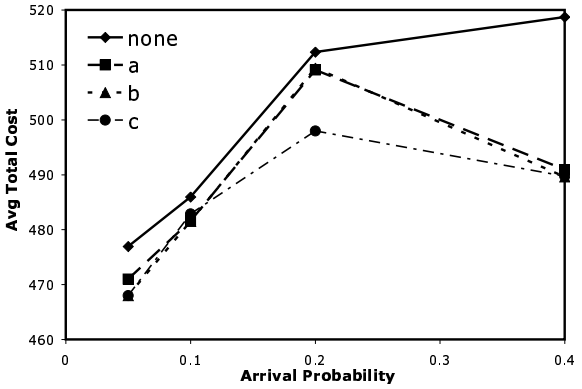

Figure 3: Average total costs for bupa with $\tau = 4$

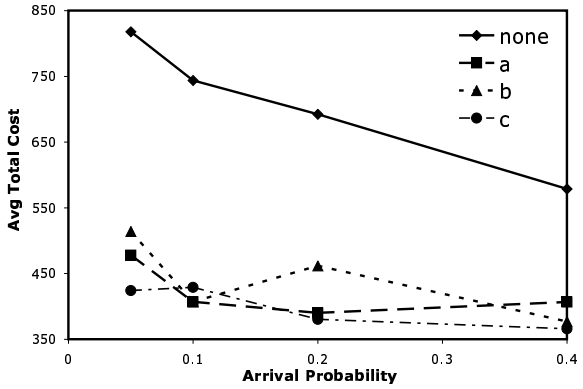

Figure 4: Average total costs for pima with $\tau = 8$]

data set has nine attributes in 683 instances (444 negative, 239 positive). The pima data has eight attributes in 768 instances (500 negative, 268 positive). The bupa data set has five attributes in 345 instances (169 negative. 176 positive).

These data sets all have associated attribute measurement costs, but in these experiments we set w_M to zero so as to ignore attribute measurement costs, as time and MC costs are the primary focus of this study. For each data set the attributes with highest information gain (seven in breast, six in pima, four in bupa) are given time measurement distributions that are normal with mean 6 and standard deviation 2. The remaining attributes can be measured instantly. This assignment of time does not reflect the actual time to measure for these attributes, which for these data sets are unknown. In lieu of that knowledge, we can most effectively study the tradeoff between misclassification and time costs by imposing measurement times on the most predictive attributes. The time cost weight w_T is set to 1. The time cost function $C_T(t) = t^2$. Time is discretized using the method discussed in Section 3.4 on the single instance problem. We use $\tau = 8$ for pima and breast, and $\tau = 4$ for bupa.

MC costs are set up for each instance so that correct predictions have penalty zero, the more frequent class A has penalty 1, and the less frequent class B has misclassification cost penalty $(|A|/|B|)^2$. This reflects the fact that in most CSC problems, the rarer class is more costly to label incorrectly. Misclassification cost weights are set to 1000.

Arrival of new instances in the queue at each time unit is sampled from a Bernoulli distribution where a single instance arrives at each time unit with probability a. This parameter is varied to simulate a variety of loads on the sequential classifier: $a = 0.05, 0.1, 0.2, 0.4$. When solving the MDP using $\tau > 1$, the probability of n instances arriving during the unit of τ is computed from the binomial distribution $P_a(n|\tau)$. The resolution of the queue average waiting time parameter was set to τ, and the number of instances parameter to $\tau/2$.

Five fold stratified cross validation was performed on each data set/arrival rate pair, with a 2/3 vs. 1/3 split between training and testing data.

5.2 Results

Figures 3, 4, and 5 show the average total cost per instance incurred in each of the three data sets for each of the four instance arrival rates. Figure 5 shows a typical breakdown between time and MC costs. The labels 'a', 'b', and 'c' represent the OC estimation methods presented in equations 3, 5, and 6 respectively, while 'none' indicates the performance of the single instance policy that is ignorant of waiting instances. We see that estimating the opportunity cost results in lower cost policies in all data sets and all arrival rates than following the single instance policy. There is no clear winner among the three OC approximation schemes.

One obvious question is why didn't method 'c' perform better? This stems from the fact that the estimated cost distributions in method 'c' are computed using a single instance policy. The distribution of costs in the single instance policy is likely to be significantly different from the distribution that would be seen given the current state of the queue. Costs in the current instance are likely to be larger (due to MC costs) when many instances are waiting. The costs from the single policy distribution are thus often too small, resulting in substantial underestimates of the actual OC. Further research will address this shortcoming by iteratively re-estimating the cost distribution: the policy π_c is first computed using cost and time distributions from the single instance policy, as usual. We can then solve for a new policy using cost and time distributions estimated from π_c. This process can iterate until the policy no longer changes from iteration to iteration.

Additionally, we can observe evidence of overfitting in the bupa and pima cases. For example, the single instance policy on pima incurs lower costs as the arrival rate increases. Because queued instances are ignored, start times for individual instances increase with arrival rate. The policies for higher start times measure less instances, which results in smaller MC costs on test data versus policies that measured many attributes and overfit the training data. Overfitting must be given careful consideration in further research.

6. CONCLUSIONS

Existing cost-sensitive classification algorithms have focused solely on misclassification and attribute measurement costs. Yet for many applications, good responsiveness is a desirable and often necessary property. In this research we have shown novel methods for dealing with time sensitive classification problems in both the single instance and

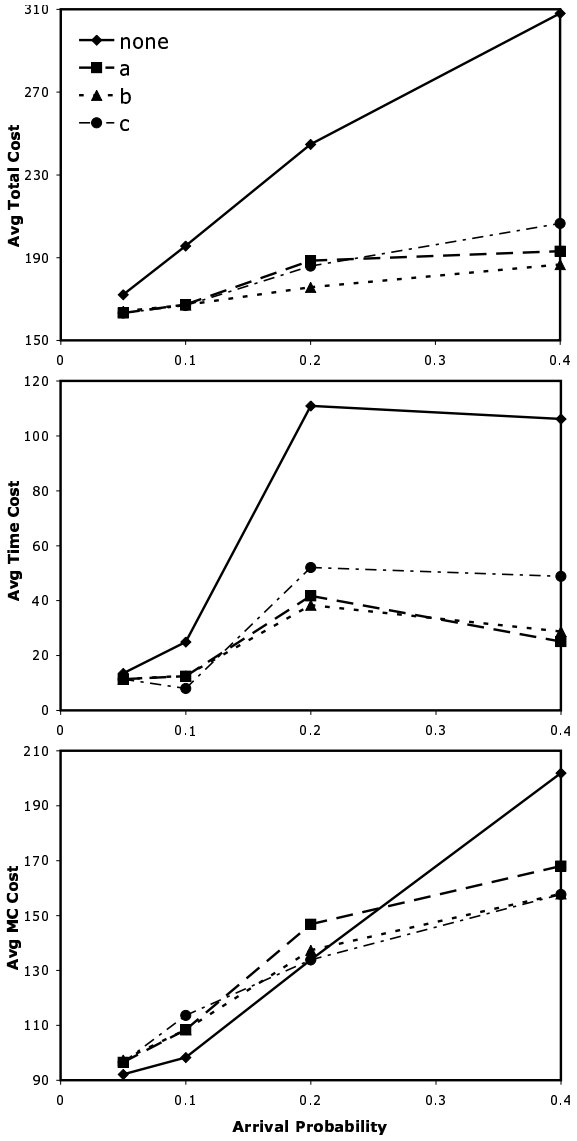

Figure 5: Costs for breast with $\tau = 8$

sequential cases. By intelligently discretizing time and effectively approximating opportunity costs, policies can be computed for a wide range of classification problems.

Avenues for further research involve relaxing the attribute measurement model. These include allowing for stochastic attribute values and repeated measurement of said attributes. A differentiation should also be made between 'blocking' and 'nonblocking' measurement actions. During a blocking measurement, the processor is busy performing the measurement computations. However, during a nonblocking measurement, such as retrieving a document over the Internet, other measurements (including those in subsequent instances) can simultaneously be performed.

7. ACKNOWLEDGMENTS

Support for this work was provided in part by the National Science Foundation under Grant No. 0328601.

8. REFERENCES

[1] A. Arnt and S. Zilberstein. Learning to perform moderation in online forums. In *Proc. IEEE/WIC Intl. Conf. on Web Intelligence*, 2003.

[2] A. Arnt and S. Zilberstein. Attribute measurement policies for time and cost sensitive classification. In *Proc. 4th IEEE International Conference on Data Mining (ICDM'04)*, pages 323–326, 2004.

[3] A. Arnt, S. Zilberstein, J. Allan, and A. I. Mouaddib. Dynamic composition of information retrieval techniques. *Journal of Intelligent Info. Systems*, 2004.

[4] C. Blake and C. Merz. UCI repository of machine learning databases, 1998.

[5] B. Bonet and H. Geffner. Learning sorting and decision trees with POMDPs. In *Proc. 15th Intl. Conf. on Machine Learning*, pages 73–81, 1998.

[6] X. Chai, L. Deng, Q. Yang, and C. X. Ling. Test-cost sensitive naive bayes classification. In *Proc. 4th IEEE International Conference on Data Mining (ICDM'04)*, pages 51–58, 2004.

[7] P. Domingos. Metacost: A general method for making classifiers cost-sensitive. In *Proc. 5th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 155–164, 1999.

[8] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan. AdaCost: misclassification cost-sensitive boosting. In *Proc. 16th Intl. Conf. on Machine Learning*, pages 97–105, 1999.

[9] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proc. 13th Intl. Joint Conf. on Articial Intelligence*, pages 1022–1027, 1993.

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

[11] A. Guo. Decision-theoretic active sensing for autonomous agents. In *Proc. 2nd Intl. Conf. on Computational Intelligence, Robotics, and Autonomous Systems*, 2003.

[12] E. Horvitz and G. Rutledge. Time-dependent utility and action under uncertainty. In *Proc. 7th Conf. on Uncertainty in Artificial Intelligence*, pages 151–158, 1991.

[13] N. J. Nilsson. *Principles of artificial intelligence*. Tioga Publishing Company, 1980.

[14] E. Pednault, N. Abe, and B. Zadrozny. Sequential cost-sensitive decision making with reinforcement learning. In *Proc. 8th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 259–268, 2002.

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

[16] V. B. Zubek. Learning diagnostic policies from examples by systematic search. In *Proc. 20th Conf. on Uncertainty in Artificial Intelligence*, pages 27–34, 2004.

[17] V. B. Zubek and T. Dietterich. Pruning improves heuristic search for cost-sensitive learning. In *Proc. 19th Intl. Conf. on Machine Learning*, pages 19–26, 2002.