Attribute Measurement Policies for Cost-effective Classification

Andrew Arnt* Shlomo Zilberstein†

Abstract

Many systems with machine learning classifiers as components require the ability to function in a realtime, online setting. Such systems must be able to quickly classify instances so as to minimize a variety of costs. We identify three components of cost that must be considered: penalties incurred due to the misclassification of an instance, costs incurred when measuring an attribute of the instance, and a utility cost related to the time elapsed while measuring attributes. We show how to model this problem as an Markov Decision Process (MDP), and then use AO* heuristic search to build a policy given a set of labeled training data. Additionally, we discuss how to modify this system to cope with a *stream* of instances arriving over time, where time taken to measure attributes in the current instance can influence time-sensitive costs of waiting instances.

1 Introduction

Machine learning classifiers predict labels for instances using many attributes measured or computed from the instance. Some of these attributes may be computationally intensive to compute or rely on relatively slow external sources of information. It may be impractical or even infeasible to measure all possible attributes for each instance when in a realtime setting. To address these issues, we develop a model that allows the system to quickly decide which attributes to measure, what order to measure them in, and when to stop attribute 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.

The next section describes the three types of cost that can be incurred by a classifier operating in a time sensitive environment. We note that previous work has only dealt with two of these costs. A simple greedy, non-optimal approach to developing attribute measurement and classifcation policies is described in Section 3. We show how this problem can be formulated as an MDP and solved using AO* search in Section 4. Finally, an approarch to handling a sequence of classification tasks is discussed in Section 5.

*University of Massachusetts, Amherst
†University of Massachusetts, Amherst

2 Cost Functions

The cost function C is designed so that minimizing C will in turn cause the system to provide the highest quality of service. As such, there are three components that reflect desired system properties.

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 as having a predicated label l_p. Minimizing this component will minimize the severity and frequency of classification errors.

The relative costs of classification errors are often represented in a cost matrix. An example cost matrix for an email filtering program whose task is to detect and automatically delete 'spam' in shown Table 1. When a false positive occurs, a legitimate email flagged is spam, and the consequences may be dire: some possibly important message has been deleted unread. A false negative is a spam message that slips through the filter, and is often just an annoyance to the user. Therefore false positives are deemed 10 times more costly than false negatives. In medical diagnosis, a false negative on some tests can cause big problems for the patient being diagnosed. However a false positive may result in unnecessary treatment.

The misclassification cost component depends on the actual label l_a of an instance, which is unknown, unless the instance is part of a set of labeled training data. Therefore, in practice we need to use the *expected* misclassification cost, given that the classifier predicts label l_p:

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

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 $p(l_a|\mathbf{f})$, and must be estimated 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 examples such that for every measured attribute in $\mathbf{f}$, the training example has the same value, and $train(l_a, \mathbf{f})$ is the subset of examples in $train(\mathbf{f})$ that have label l_a. These probabilities can be smoothed using a Laplace correction.

The label l_p that is predicted for the current attribute vector $\mathbf{f}$ can come from one of two sources. In the first case, l_p is the output corresponding to the input $\mathbf{f}$ by a classification algorithm that has been previously trained on some training data set. The classification algorithm can be any model: naive Bayes, decision tree, nearest neighbor, etc. so long as the classification procedure is robust with respect to missing values in the attribute vector. There are a variety of methods for handling missing values as discussed, for example, in these works: [10, 9, 12]. It may advantageous to use a training algorithm that is tuned to minimize classification costs, instead of one that tries to maximize classification accuracy.

Alternatively, we can build the choice of l_p into the algorithm itself. That is, the algorithm will choose the label that incurs the minimum expected misclassification cost on a set of training data:

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

There has been a significant volume of work on the problem of minimizing misclassification costs: some general methods are the weighted boosting algorithm of [5], and the MetaCost algorithm of [4].

2.2 Attribute Measurement Costs An individual attribute f_i may have a fixed, deterministic cost to measure: $C_M(\mathrm{m}(f_i))$. For example, this could correspond to having to pay to retrieve some piece of information necessary to compute an attribute. In the field of medical diagnosis, it is very common for some tests to have significant financial costs. Research that handles both attribute measurement costs and misclassification costs includes the genetic algorithm based decision tree inducer of [11], the POMDP (Partially Observable Markov Decision Process)-based decision tree learner of [2], a dynamic programming algorithm described in [6], a POMDP for computing attribute measurement policies with respect to a given Naive Bayes classifier in [7], and finally an MDP framework with heuristic search to find good attribute measurement and classification policies [13].

Table 1: Example cost matrix for an email classification task

		l_a	
	$C_L(l_p\|l_a)$	ok	spam
l_p	ok	0	1
	spam	10	0

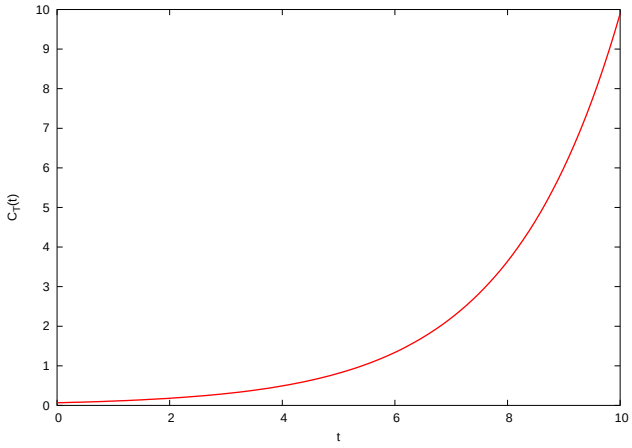

Figure 1: Example time sensitive component of the cost function, $C_T(t)$

2.3 Response Time Costs The cost function should also reflect the timeliness with which we wish the classifier to act. In many systems 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 try to be fairly responsive and not spend unreasonable amounts of time 'thinking' about the correct label of an instance. Many embedded or resource limited systems are used in environments where responsiveness is a key property of the system.

Therefore the cost function has a final component $C_T(t)$, which will typically have the form shown in Figure 1. Note that the cost of a quick result is small and fairly constant, but as the waiting time increases, the cost grows at an increasing rate. This is a good model of user's perceived utility of a system when they are forced to wait for a result. In general, however, the time cost component can take on any form, so long as the time cost is nondecreasing over time.

It is important to note that none of the research referenced in Sections 2.1 or 2.2 allowed for costs based on the time required to classify an instance. Costs in those models are incurred only by the direct, immediate

cost of measuring an attribute or by misclassifications.

With some constraints, it is possible to represent time-sensitive classification costs as immediate deterministic measurement costs of individual attributes. Given these constraints, the above methods are suitable. First, the time to measure attributes must be deterministic. Secondly, the time-sensitive portion of the cost function, $C_T()$ must be linear over time. If this is the case, attribute f_i that takes exactly δ time units to compute, can be given an augmented measurement cost that includes the incremental time cost incurred:

$$C_{M'}(\mathrm{m}(f_i)) = \delta \left(\frac{d}{dt} C_T()\right) + C_M(\mathrm{m}(f_i))$$

instead of using a final time-dependent cost incurred upon classification. Both of these constraints are impractical for real life systems. Any attribute that takes time to compute or retrieve will rarely have a deterministic time to measure. Additionally, a linear time cost function is a poor representation of a human's value of time. A linear function implies that a user loses as much value per time unit for very short wait time as for longer wait times. In reality, users typically have some patience initially, and become increasingly frustrated as the wait increases. This behavior is best reflected by a cost function with a derivative that increases over time.

2.4 Combining the cost function components

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

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

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

3 Myopic attribute measurement policies for single tasks

For the task of classifying a single instance, starting with no measured attributes, we want to iteratively choose to measure the single attribute that has the largest value of information (VOI) [8]. This is equivalent to finding the attribute that maximizes the one step reduction in the expected cost (EC) of immediate classification. The current state of the computation is $s = \langle \mathbf{f}, t \rangle$, where the current attribute vector for the instance is $\mathbf{f}$ and the current waiting time of the task is t. Note that $\mathbf{f}$ may contain unmeasured attributes.

The VOI of measuring an attribute f_i is the expected decrease in cost between when the state f_i is not measured and the state when it is measured.

$$VOI(\mathrm{m}(f_i)|s) \quad = \quad E(C(s) - C(s'))$$

The state s' is the new state after f_i has been measured, taking a duration of δ time units: $s' = \langle \mathbf{f} \cup f_i = x, t+\delta \rangle$, where $\mathbf{f} \cup f_i = x$ refers to $\mathbf{f}$ with attribute f_i set to x. We assume the measured value of an attribute and the time it takes to measure are independent. The expected difference in cost before and after f_i is measured is the sum of the immediate measurement cost of attribute f_i and the differences between the new and current expected misclassification costs and time penalties (for clarity, we omit the cost weight parameters):

$$\begin{aligned} E(C(s) - C(s')) = C_T(t) + EC_L(\mathrm{cl}(l_p)|\mathbf{f}) - C_M(\mathrm{m}(f_i)) - \\ \sum_{\delta \in T_i} C_T(t+\delta) p(T_i = \delta) - \\ \sum_{x \in f_i} EC_L(\mathrm{cl}(l'_p)|\mathbf{f} \cup f_i = x) p(f_i = x|\mathbf{f}) \end{aligned}$$

where T_i is the set of all possible durations that attribute f_i can take to measure. The probability that attribute f_i will take on value x given the incomplete attribute vector $\mathbf{f}$ is $p(f_i = x|\mathbf{f})$ and must be estimated from training data. The probability of attribute f_i having value x given the already measured attributes in $\mathbf{f}$ is

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

These probabilities can also be smoothed with a Laplace correction.

The probability that attribute f_i takes δ time units to measure is denoted as $p(T_i = \delta)$. This value must also be estimated from a set of training data or from some other source of prior experience. Note that this probability is not conditional on $\mathbf{f}$, even though the measurement of previous attributes may affect the time to measure f_i. For example, if an attribute that requires information to be downloaded from the Internet has already been measured, then any remaining attributes that depend on that same information can retrieve it from a fast local cache. In general, when attributes i and j depend on the same information source or computational result and $p(T_j = 0|\mathbf{f} \cup \mathrm{m}(f_i)) = 1$ (or vice versa), these attributes should be merged into a single attribute taking on the value of all pairs $\langle i, j \rangle$ when running the attribute selection algorithm.

If none of the measurements reduce the cost ($\forall f_i \, VOI(\mathrm{m}(f_i)|s) \leq 0$), the instance is classified using the current incomplete attribute vector $\mathbf{f}$ to determine l_p. If all attributes have already been measured, then the next action will be to classify the instance. This combined attribute measurement and classification policy can be computed off line and then later be quickly followed in an on line setting.

Using myopic attribute selection, the system will greedily try to minimize cost by extracting the single attribute that provides the largest cost reduction. While this procedure is non-optimal, this myopic approach may be enough to give sufficient results, without the complexity that would be needed to compute an optimal attribute measurement policy. Optimal methods are discussed in the next section.

4 Optimal cost sensitive learning

In the above sections, attributes were selected for measurement one at a time, greedily trying to minimize the expected cost. However, to find the optimal measurement policy, a more sophisticated algorithm than a one step look-ahead is necessary.

This optimal online attribute measurement problem is to find the attribute measurement policy that, given a set of training data, 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 learning builds on the work of Zubek and Dietterich in [13]. We frame the attribute measurement and classification problem as an MDP (Markov Decision Process). The "optimal" policy (quoted because it is only optimal with respect to a set of labeled training data) can then be found using AO* search, a classical heuristic search technique. We extend the model in [13] to handle time-sensitive utility costs.

We will examine both the case where classification labels are chosen to minimize expected cost within the search itself (as is the case in [13]), and also the case where classifications are made by an external misclassification cost sensitive classifier.

4.1 MDP overview A Markov Decision Process (MDP) is 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 state effectively summarizes all previous activity of the agent in the environment. The mapping from states to actions that maximizes reward is called the *optimal policy*.

4.2 Time and cost sensitive classification as an MDP The states $s \in S$ in the model presented in [13] are simply the set of all possible attribute vectors $\mathbf{f}$, including those with unmeasured attributes. To accommodate time sensitive costs, the state space must be augmented to include the current waiting time of the task: therefore the state space is $s = \langle \mathbf{f}, t \rangle$ for all values of $\mathbf{f}$ and t (time is discretized and bounded empirically to provide good performance at an acceptable level of computational complexity), in addition to an additional absorbing termination state E that is transitioned to when an instance is classified. The starting state of the MDP is the state with no measured attributes and no elapsed waiting time:

$$s = \langle \mathbf{f} = \langle f_1 =?, f_2 =?, \ldots, f_{|\mathbf{f}|} =? \rangle, t = 0 \rangle$$

.

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 three types of cost related to taking an action. $C_M(\mathrm{m}(f_i))$ is the cost to measure attribute f_i. This is a deterministic quantity, unrelated to time. There are also incremental time costs $C_\Delta(\delta|t)$ which indicates 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 $C_T(t)$ such as the one shown in Figure 1, 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 $C_M(\mathrm{m}(f_i)) + \sum_{\delta \in T_i} p(T_i = \delta) C_\Delta(\delta|t)$.

We assume the measured value of an attribute and the time it takes to measure are independent, so the probability of transitioning from state $s = \langle \mathbf{f}, t \rangle$ to state $s' = \langle \mathbf{f} \cup f_i = x, t+\delta \rangle$ is

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

The probability of arriving in terminal state E when the classification action is taken is always one:

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

4.3 AO* Search AO* search is an heuristic search algorithm for searching AND/OR graphs. 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 to compute the best action.

AO* works by iteratively improving upon the current best partial solution policy until the optimal policy is found. Each iteration of the AO* search is composed of two parts. In the first part, the current best partial solution is expanded (meaning it's 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 during the search is $f(s)$ and is an optimistic estimate of the cost to get from state s to a terminal state.

Note that AO* search does not have the ability to find policies containing loops. This is not a problem for this domain, as the underlying MDP does not allow for the same state to be visited twice for a single instance.

4.4 Heuristics For AO* search, a heuristic is necessary to guide the search. 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. [13] uses an optimistic one-step lookahead. We use the same heuristic with the inclusion of incremental time costs. Given an unexpanded state s, the heuristic value $h(s)$ is the cost of the action (classifying or measuring an attribute) giving the smallest immediate cost:

$$h(s) = \min_{f_i \notin \text{meas}(\mathbf{f})} \begin{cases} EC_L(\text{cl}(l_p)|\mathbf{f}) \\ C_M(\text{m}(f_i)) + \sum_{\delta \in T_i} p(T_i = \delta) C_\Delta(\delta|t) \end{cases} \tag{4.1}$$

4.5 Evaluation function decomposition and weighted heuristics Chakrabarti et al [3] show that the evaluation function in AO* search can be decomposed into $f(s) = g(s) + h(s)$, similar to the decomposition seen in A* search. However, the meanings of these functions differ. In A* search, $g(s)$ represents the cost to arrive at state s from the start state, and $h(s)$ is the heuristic estimate of how much cost remains to go from s to a goal state. However, in AO*, $g(s)$ represents the portion of the solution cost of the policy rooted at s that is *known*; That is, the $g(s)$ cost has been explicitly computed using just costs incurred on transitions to generated states, with no $h()$ values being considered. The $h(s)$ value represents the costs in the policy rooted at s that are just estimated using the h estimates at unexpanded nodes.

The functions g and h are both computed in a bottom-up fashion. Generated but not yet expanded states s have $g(s) = 0$, and $h(s)$ as defined in Equation 4.1.

If the current best action at expanded state s is to classify, then the heuristic cost estimate will be zero, because the only future costs incurred will be the known expected misclassification cost:

$$\begin{aligned} g(s) &= EC_L(\text{cl}(l_p)|\mathbf{f}) \\ h(s) &= 0 \end{aligned}$$

If the best action at state s is to measure attribute f_i then we have

$$\begin{aligned} g(s) &= C_M(\text{m}(f_i)) + \\ &\quad \sum_{\delta \in T_i} p(T_i = \delta) \left(C_\Delta(\delta|t) + \sum_{x \in f_i} p(f_i = x|\mathbf{f})\, g(s') \right) \\ h(s) &= \sum_{\delta \in T_i} p(T_i = \delta) \sum_{x \in f_i} p(f_i = x|\mathbf{f})\, h(s') \end{aligned}$$

where $s' = \langle \mathbf{f} \cup f_i = x, t + \delta \rangle$.

This decomposition of the value function allows a weighted version of AO* that gives more emphasis to the heuristic estimate. Instead of using $f(s) = g(s) + h(s)$, the estimated cost can be given more weight:$f'(s) = (1 - \omega)g(s) + \omega\, h(s)$. As ω increases from 0.5 to 1.0, the search becomes increasingly greedy, producing a policies of decreasing quality. The search space explored also shrinks, allowing a tradeoff between quality of results and computability for large search spaces. This results in a bounded loss of optimality, but also can significantly reduce the number of search states that must be evaluated.

4.6 Pruning strategies For classification problems where instances have a large number of measurable attributes, each of which can take on many values, pruning of 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 by from any state of this MDP by taking a single classification action [13]. This property, which is not applicable for general MDP models, allows for some significant pruning of the search space. An upper bound $\hat{h}(s)$ value is computed at each node; this value represents the expected cost of following the current *best known* policy from search state s. Formally, $\hat{h}(s)$ is

computed as:

$$\hat{h}(s) = \begin{cases} \infty & s \text{ not expanded} \\ \hat{h}'(s) & s \text{ expanded} \end{cases}$$

$$\hat{h}_f(\mathrm{m}(f_i)|s) = C_M(\mathrm{m}(f_i)) + \sum_{\delta \in T_i} p(T_i{=}\delta)(C_\Delta(\delta|t) + \sum_{x \in f_i} p(f_i{=}x|\mathbf{f})\,\hat{h}(s'))$$

$$\hat{h}'(s) = \min_{f_i \notin \mathrm{meas}(\mathbf{f})} \begin{cases} EC_L(\mathrm{cl}(l_p)|\mathbf{f}) \\ \hat{h}_f(\mathrm{m}(f_i)|s) \end{cases}$$

again with $s' = (\mathbf{f} \cup f_i = x, t + \delta)$. Therefore, any unexpanded search node s' with parent node s where $\hat{h}(s) < h(s')$ 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 $\hat{h}(s)$.

[13] also examines statistical pruning of the search space due to sparsities of training data. This is an important consideration, since the amount of training data that is being used to direct the search gets smaller and smaller as measurement actions are taken. Eventually, the search reaches a depth where there is insufficient training data to justify choosing any one action over any other. A statistical test can be performed prior to node expansion to determine if the expansion of the node is justified by the training data matching that node. This pruning has two benefits: the search space is reduced, and more importantly, overfitting of the training data is avoided.

5 Attribute measurement for sequences of tasks

The above procedures do not account for other tasks that need to be classified. Indeed, suppose that instead of a single classification task to process, the system has to handle a *stream* of classification tasks arriving over time. Therefore, when deciding which attributes to measure in the current task, we must also consider the potential for utility loss due to delay in processing of all other tasks waiting to be classified. Arnt et al refer to this as the *opportunity cost* [1], the loss of expected value due to delay in the starting of work on the remaining tasks. 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 tasks, such as number of tasks waiting and average waiting time of each task. Call this vector $\mathbf{q}$.

We can factor in opportunity cost to the myopic attribute measurement strategy discussed in Section 3 by amending the expected different in cost between two states to include a component $C_{OC}(\delta|\mathbf{q})$ representing the cost of delaying computation for δ time units given a queue of waiting classification tasks represented by attribute vector $\mathbf{q}$.

In the MDP model for optimal policy construction, the opportunity cost component can be introduced by augmenting the incremental time cost component $C_\Delta(\delta|t)$:

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

This requires that the state space be expanded to $s = \langle \mathbf{f}, t, \mathbf{q} \rangle$. Therefore it may be necessary to severely limit the possible values of $\mathbf{q}$ to avoid an explosion in the overall size of the state space. For example, the task queue may be represented with just three states corresponding to the overall frequency of task arrival: *high*, *medium*, and *low*.

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. If the arrival rate of tasks in the queue is very steady and regular, then the probability mass for this distribution will be tightly concentrated around a single $\mathbf{q}'$. If the arrival rate is erratic and very unpredictable, then the probability mass will be more evenly distributed. Note that if we wished to consider such things as the time of day, or long term trends in arrival rate, these must be represented in $\mathbf{q}$. Such a complex representation of the task queue will cause the overall state space to be prohibitively large, so we will not consider these indicators.

6 Conclusions

The challenge of classification under time pressure has not seen much attention in research, with many algorithms and methods focusingly solely on misclassification and attribute measurement costs. Yet for many systems, good responsiveness is a desirable and sometimes necessary property. The problem of time sensitive classification is further compounded when dealing with a stream of instances to be classified. If future instances are not considered while choosing which attributes to measure for the current task, there is the risk of large penalties for the delay in classifying those future tasks.

This research is just underway and thus there are no experimental results to report. However, given previous work we are confident of the success of these methods. Mapping the problem without time costs to an MDP

and solving using AO* search has shown success in [13]. Furthermore the opportunity cost method for processing streams of tasks has been demonstrated in [1].

Possible future work on this subject includes allowing for the fact that attribute measurement may fail on occasion, or that the attribute values themselves are stochastic and repeated measurements may be necessary to get an accurate value.

References

[1] Andrew Arnt, Shlomo Zilberstein, James Allan, and Abdel Illah Mouaddib. Dynamic composition of information retrieval techniques. *Journal of Intelligent Information Systems*, to appear.

[2] Blai Bonet and Héctor Geffner. Learning sorting and decision trees with POMDPs. In *Proc. 15th International Conf. on Machine Learning*, pages 73–81. Morgan Kaufmann, San Francisco, CA, 1998.

[3] Partha P. Chakrabarti, Sujoy Ghose, and S.C. DeSarkar. Admissibility of AO* when heuristics overestimate. *Artificial Intelligence*, 139(2):137–174, 2002.

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

[5] Wei Fan, Salvatore J. Stolfo, Junxin Zhang, and Philip K. Chan. AdaCost: misclassification cost-sensitive boosting. In *Proc. 16th International Conf. on Machine Learning*, pages 97–105. Morgan Kaufmann, San Francisco, CA, 1999.

[6] Russell Greiner, Adam J. Grove, and Dan Roth. Learning cost-sensitive active classifiers. *Artificial Intelligence*, 139(2):137–174, 2002.

[7] AnYuan Guo. Decision-theoretic active sensing for autonomous agents. In *Proceedings of the 2nd International Conference on Computational Intelligence, Robotics, and Autonomous Systems*, 2003.

[8] Ronald A. Howard. Information value theory. *IEEE Transactions on Systems Science and Cybernetics*, SSC-2:22–26, 1966.

[9] Wei Zhong Liu, Allan P. White, S. G. Thompson, and M. A. Bramer. Techniques for dealing with missing values in classification. *Lecture Notes in Computer Science*, 1280:527–536, 1997.

[10] J. Ross Quinlan. Unknown attribute values in induction. In *Proc. 6th International Workshop on Machine Learning*, pages 164–168, 1989.

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

[12] Zijian Zheng and Boon Toh Low. Classifying unseen cases with many missing values. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 370–374, 1999.

[13] Valentina Bayer Zubek and Thomas Dietterich. Pruning improves heuristic search for cost-sensitive learning. In *Proc. 19th International Conf. on Machine Learning*, pages 27–34. Morgan Kaufmann, San Francisco, CA, 2002.