Reinforcement Learning for Active Model Selection

Aloak Kapoor
aloak@cs.ualberta.ca

Russell Greiner
greiner@cs.ualberta.ca

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

ABSTRACT

In many practical Machine Learning tasks, there are costs associated with acquiring the feature values of training instances, as well as a hard learning budget which limits the number of feature values that can be purchased. In this budgeted learning scenario, it is important to use an effective "data acquisition policy", that specifies how to spend the budget acquiring training data to produce an accurate classifier. This paper examines a simplified version of this problem, "active model selection" [10]. As this is a Markov decision problem, we consider applying reinforcement learning (RL) techniques to learn an effective spending policy. Despite extensive training, our experiments on various versions of the problem show that the performance of RL techniques is *inferior* to existing, simpler spending policies.

Categories and Subject Descriptors

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

General Terms

Algorithms

Keywords

budgeted learning, data acquisition, learning costs, training costs

1. INTRODUCTION

Traditional learning theory assumes the existence of a sufficient number of training examples for learning the target concept. In practice, however, there are often costs for acquiring the value of each feature for each training example. Here, the actual quantity of training data is limited by the learner's finite budget for purchasing features. In this budgeted learning scenario, it is critical to develop an intelligent purchasing policy which collects feature values in a way that maximizes the expected accuracy of the resulting classifier.

UBDM August 21, 2005, Chicago, Illinois, USA

To gain insight into the budgeted learning problem, we examine a simpler variant known as *active model selection*: Use a set of b "probes" to determine which of a given set of n objects has the highest expected value, where each probe of a specified object returns a value drawn from that object's distribution. Here we need to identify a strategy for deciding when to probe each object, based on the results of the previous probes and any prior knowledge of the objects. After exhausting the budget of b probes,[1] we select a single object to return, and receive a reward corresponding to the difference between true expected values of the object returned and the best possible object — i.e., the one with the highest expected value. This formulation allows pure exploration of the objects with the budget, as it delays all reward until the final time step. In addition, this budgeted problem accurately captures the training phase of budgeted learning in general, in which features of labelled training instances can be purchased in any way, with a single one-time reward (i.e., the classification accuracy) being received once the budget is exhausted and the final learned classifier is applied. Moreover, previous research has shown that policies that perform well on active model selection translate directly to effective spending policies for the budgeted learning problem. As a result, active model selection can be used as a way to prototype the performance of strategies for general budgeted learning.

Our focus in this paper is on applying reinforcement learning (RL) techniques to the active model selection problem. Although the existing (heuristic) purchasing policies for active model selection perform well, we demonstrate in this work that they still leave room for improvement. We also show that RL is well-suited to the task because (from a Bayesian viewpoint) the active model selection problem is a finite Markov Decision Process with a completely known environment model. We proceed to learn various purchasing policies with RL, where each policy is learned using a different set of features for function approximation. Despite the variety of feature sets tested, we observe that the policies produced by reinforcement learning are inferior to existing, simpler policies. This provides evidence that RL techniques, if they can be applied at all, will require a more sophisticated choice of features in order to be effective for active model selection, and hence for the more complex problem of budgeted learning.

The remainder of the paper is organized as follows. Sec-

[1] Actually, we can assume different objects have different probe costs; we require only that the total cost of the probes be under b. See Section 2.

tion 2 presents the active model selection problem formally, and introduces some of the simple purchasing policies from previous research. Section 3 frames the problem as a Markov Decision Process and discusses the reinforcement learning techniques that we will use. Section 4 provides empirical results comparing RL to the existing policies. In closing, Section 5 discusses related literature and Section 6 provides conclusions and a discussion of future work. All proofs can be found in the Appendix.

We will assume the reader is already familiar with the basic notions of reinforcement learning; if not, we refer the interested reader to [17].

2. ACTIVE MODEL SELECTION

The input to the problem is:

- A set of n independent Bernoulli random variables $\{C_1, \ldots, C_n\}$ with unknown success probabilities. For simplicity of exposition, we can think of these C_i as a set of coins, where the unknown success probability is the probability of the coin turning up heads when flipped.
- A set of n prior distributions (i.e., density functions) over the head probability of each coin C_i. That is, the *head probability* of each coin C_i is itself treated as a random variable X_i, and a prior density function $f(X_i)$ is provided as a distribution over the possible head probabilities of coin C_i.
- A set of n (known) costs $\{S(C_i)\}$ for flipping the coins.
- A finite (known) budget $b \geq 0$ that can be spent flipping the coins.

Given these inputs, the active model selection problem proceeds as follows. Any coin C_i can be flipped at any time, as long as the remaining budget, denoted by b', satisfies $b' \geq S(C_i)$. We use the outcome of each coin flip to update the density function for the flipped coin. For example, if coin C_i is flipped and turns up heads, then its density function is updated to $f(X_i|C_i = heads)$; of course, a similar update occurs for a tails outcome. (We describe the exact format of the density function and the updates in our simplifying assumptions below.) Coin flips and density updates continue until the budget is exhausted ($b' = 0$). Once the budget is exhausted, the learning period is over, and a single coin must be chosen — this coin C^* (and only this coin) will be used in all future flips, for which we will receive rewards for head outcomes. Of course, even when $b' = 0$, we will still not *know* the true head probability for this (or any) coin, and so will not know whether coin C^* actually has a better head probability than the other coins. The best we can do is to choose the coin that minimizes our future regret of selecting it. To do this, we define a new random variable X_{max} to be the maximum head probability over all of the coins: $X_{max} = max_i(X_i)$, and now the Bayesian regret of choosing coin C_i is:

$$\text{Regret}(C_i) \;=\; \int_{\vec{X}} (X_{max} - X_i) \prod_{j=1}^{n} f(X_j)\; d\vec{X} \qquad (1)$$

Notice that we minimize regret by choosing the coin whose mean head probability $E(X_i)$ is largest [10]. Let this maximum mean coin be $C^* = \arg\max_{C_i} E[X_i]$. Thus, when the budget is exhausted, C^* *should* be selected. (We will later use $C^*(\mathbf{o}) = \arg\max_{C_i} E[X_i|\mathbf{o}]$, as a function of the observed coin flip outcomes $\mathbf{o}$.)

Before introducing the overall (regret-related) objective function we wish to minimize, we must first introduce the notion of a policy. A policy π for active model selection specifies which coin to flip at each time step. Formally, a policy is a mapping π: $\langle\, b',\ f(X_1),\ \ldots,\ f(X_n)\,\rangle \rightarrow [1, n]$ that specifies the index of the coin to flip, given the current state defined by the remaining budget and the posterior distributions over the coins.

Since the result of every coin flip is stochastic, a policy for flipping the coins can result in *several* different "outcome" states in which the budget is exhausted. Thus, a policy π for active model selection is scored based on its expected regret:

$$\text{ER}(\pi) \;=\; \sum_{\mathbf{o} \in outcomes(\pi)} P(\mathbf{o})\ \text{Regret}(C^*(\mathbf{o})) \qquad (2)$$

where the sum is over the various "outcomes" of the policy when the budget b' has been exhausted. The objective of active model selection is to find the optimal policy π^* that minimizes Equation 2.

Conveniently, minimizing Equation 2 is equivalent to maximizing the expected head probability of the chosen coin:

$$\pi^* \quad = \quad \arg\max_{\pi} \sum_{\mathbf{o} \in outcomes(\pi)} P(\mathbf{o})\ \max_i \{E(X_i|\mathbf{o})\} \qquad (3)$$

Equation 3 is an easier objective to remember for active model selection: flip the coins so that the expected$_1$ maximum expected$_2$ head probability of the chosen coin is as large as possible. (Note that both "expected" are required as the first expectation$_1$ is over possible outcomes of the policy, while the second expectation$_2$ is over the head probability distribution of the chosen coin.)

2.1 Simplifying Assumptions

Coin C_i's head probability is represented as a random variable X_i. We assume that X_i is a Beta random variable with density function $f(X_i) = Z\ (X_i)^{\alpha-1}(1 - X_i)^{\beta-1}$ (here Z is a normalizing constant and α and β are two positive hyperparameters that define the Beta distribution).

For a Beta(α, β) distribution, the mean is $\mu = \frac{\alpha}{\alpha+\beta}$ while the variance is $\sigma^2 = \frac{\mu(1-\mu)}{\alpha+\beta+1}$. Loosely speaking, when α (β) is much larger than β (α), it means that a coin is likely to have a high (low) head probability. On the other hand, when both α and β are 1, the distribution over head probabilities is uniform.

One attractive property of the Beta distribution is that it is computationally simple to calculate posterior densities. After observing h heads and t tails on coin C_i, its posterior density is just $f(X_i|h\ \ heads,\ t\ \ tails) = Beta(\alpha + h, \beta + t)$. For example, if a coin is initially a Beta$(6, 2)$ and we observe three heads and two tails, then it becomes a Beta$(9, 4)$ coin. In some sense, the Beta hyperparameters can be viewed as simple frequency counts for a random variable with two possible outcomes.

Although the formal description allows for any coin costs, in this paper we will assume that the costs are uniform: $S(C_i) = 1\ \forall i$, and that the budget b is a positive integer. Finally, as we are studying active model selection because of its relationship to budgeted learning, we are typically in-

terested in values of b that are not much greater than n (typically $b = n \times k$, with k a small positive integer), as most budgeted learning algorithms will act reasonable when b is much larger than n. In fact, in the case where b is very large relative to n, even a simple policy (e.g. purchasing every feature of every instance) will yield a training set that can produce an accurate classifier, and so these scenarios are not of great interest from a budgeted learning point of view.

2.2 Mapping to budgeted learning

As mentioned in Section 1, active model selection is highly related to budgeted learning because it mimics the pure exploration phase (i.e., purchasing features of labelled training data), followed by the one-time reward phase (i.e., the classification accuracy of the final learned classifier). In addition to this relationship, optimal active model selection is equivalent to optimal budgeted learning of a depth-one decision tree, albeit with some rather strict assumptions. Specifically, given a binary class Y and n candidate features $\{R_i\}_{i=1..n}$ for a classification task, assume $P(R_i = 1|Y = 0) = 0 \ \forall i$. Then the best feature to use as a depth-one decision tree is: $\arg\max_{R_i} P(R_i = 1|Y = 1)$. Set coin C_i to be feature R_i, X_i to be $P(R_i = 1|Y = 1)$, and let flipping coin C_i be equivalent to purchasing feature R_i on a random $Y = 1$ instance. Then a policy π^* that maximizes the expected head probability of the chosen coin (Equation 3) also maximizes the expected accuracy of the chosen depth-one decision tree.

2.3 Standard Policies

Previous research has considered some simple policies for active model selection.

Round Robin (RR). The most intuitive algorithm is to allocate flips evenly over the coins, proceeding in a round-robin fashion. When $b = n \times k$ for an integer k, and all coins have unit cost, RR will flip each of the n coins k times. Despite its fair distribution of flips, the ratio of RR's expected regret to the expected regret of the optimal policy can be made arbitrarily large [11]. Fortunately, more effective policies than RR are known.

Biased Robin (BR). The BR algorithm repeatedly flips a coin C_i until a tail outcome occurs. Once a tail is observed, BR moves to the next coin, C_{i+1}, and repeats the process. (Of course when the last coin turns up tails, BR moves back to the first coin). This simple algorithm is well known in statistics as "play the winner" [14] and has been previously studied as a sampling method for clinical trials [6]. Its performance on the coins problem has been very strong in the case of identical starting priors. Despite its competitive performance, BR is a suboptimal policy. In fact, we can show that the number of suboptimal decisions made by BR can be made arbitrarily large:

PROPOSITION 1. *Given any positive integer $K \geq 1$, there exists a problem with $n = (K+2)$ $Beta(1,1)$ coins, and budget $b = (2n+3)$ such that the BR policy takes a suboptimal action at least K times.*

Single Coin Lookahead (SCL). The SCL algorithm computes the expected regret (Equation 2) of the policy that devotes *all* remaining flips in the budget to a single coin C_i. Whichever coin yields the policy with lowest expected regret is flipped *once*, and then SCL repeats the previous calculation with its reduced budget to choose the next coin. Like BR, SCL has strong performance, but is still suboptimal. In particular, SCL suffers in situations where multiple coins must interact heavily to produce the optimal policy. This occurs because SCL computes a score for coin C_i without considering how the remaining $n-1$ coins could interact with C_i to improve its policy. These deficiencies in the simple strategies offered by BR and SCL motivate the need for a more robust policy.

3. THE MDP FRAMEWORK

The active model selection problem is a finite-horizon Markov Decision Process [15] consisting of a set of states S, a set of actions A, a reward function R, and a transition function T. Specifically, we identify a state $s \in S$ of the MDP by the remaining budget b', and by the collection of Beta hyperparameters over the coins. That is, a state is a $2n+1$ element vector of the form: $\langle\, b', \alpha_1, \beta_1, \ldots, \alpha_n, \beta_n \,\rangle$. The complete set of reachable states corresponds to all the possible posterior Beta distributions that can occur over the n coins by spending some portion x of the original budget b, with $x \leq b$. Since no more actions can be taken once the budget is exhausted, the terminal states are those in which $b' = 0$. The actions of the MDP correspond to the n different coins that can be flipped, where action $a_i \in A$ corresponds to flipping coin i.

The reward function $R(s, a, s')$ specifies the reward of taking action a from state s and reaching state s'. In the coins problem, the reward received in any non-terminal state (i.e., where the remaining budget is positive) is zero, while the reward at a terminal state is the regret of choosing a coin C^* that might not be the best: Regret(C^*). Since our goal is to minimize the expected regret, it makes more sense to think of the rewards as costs (to be minimized) in our context.

In many MDPs, the reward at future time steps is valued less than immediate reward, and so a discount factor $\gamma \leq 1$ is used to multiply future rewards to reduce their value. In the coins problem, future rewards are no less valued than immediate rewards (in fact the *only* reward that matters is the one received on the last time step), and so we have $\gamma = 1$ in our MDP formulation.

Finally, the transition function $T(s, a, s')$ specifies the probability of reaching state s' after taking action a from state s. In our Bayesian formulation, $T(s, a, s')$ can be conveniently computed using the mean of the Beta distributions over the coins. For example, $T(Beta(4,2), C_i, Beta(5,2))$ is calculated as the expected probability of coin C_i turning up heads: $E(X_i) = 4/6$. As the transition function specifies probabilities, we often use $P(s, a, s')$ in place of $T(s, a, s')$.

3.1 The Optimal Policy

Given the MDP formulation, there are a number of techniques that can be used to solve for the optimal policy exactly [17]. For example, a bottom-up dynamic program can use the Bellman optimality equation to learn V^{π^*}, the expected value of each state under an optimal policy:

$$V^{\pi^*}(s) = \max_a \sum_{s'} P(s, a, s')[R(s, a, s') + \gamma V^{\pi^*}(s')] \quad (4)$$

Beginning at the end states in which $b' = 0$ and performing

Table 1: Feature sets used for approximating the value function

Set #	Features Groups Included In Set
1	Budget, Beta Hyperparameters
2	Budget, Means and Standard Deviations
3	Budget, Confidence Interval Stats
4	Budget, Mean Stats, Confidence Interval Stats
5	Budget, Lookahead Stats, Confidence Interval Stats

a backward sweep toward the initial state $b' = b$, the optimal value function can be completely determined. With the known transition and reward functions, the optimal policy π^* then follows immediately via greedy one-step lookahead. Unfortunately, the state space of active model selection grows exponentially with b and n, making it intractable to compute the optimal policy using exact methods such as dynamic programming. This leads to the natural alternative: approximate dynamic programming via reinforcement learning.

3.2 Applying RL to Active Model Selection

The MDP formulation coupled with the known environment dynamics (i.e., the transition and reward functions) motivates a reinforcement learning approach to active model selection. Repeated episodes of the MDP can be simulated, with an RL agent exploring the space of actions, observing the resulting rewards, and gradually learning a policy to minimize the expected regret.

For our RL investigation, we use tile-coding [17] as our function approximation method and combine it with a temporal difference (TD) learner [16] using an epsilon-greedy policy. In general the combination of epsilon-greedy TD(λ) and linear tile-coding is attractive because there are strong guarantees that the mean squared error between the approximate learned value function and the true value function will be near minimal [18]. As with any RL-agent, the number of free variables that must be manually set for a TD(λ) tile-coding agent is extensive. The large number of degrees of freedom makes any exhaustive investigation of the parameters infeasible. Still, in our experiments, we explored a wide range of points in the parameter space, including various values for the learning step-size (α), the probability of exploration (ϵ), and the weight of n-step backups (λ).

For function approximation, we collected the obvious features (e.g. the Beta hyperparameters, the remaining budget, the means and standard deviations of the coins), along with some more subtle attributes (e.g. confidence intervals, budget based confidence intervals, variation among the coins, odds ratios). Although our experiments tested numerous combinations of features, we focus here on five feature sets that are representative of the general trends we observed. For each one of the five sets, Table 1 gives the names of the different *feature groups* that are included in the set. (The interested reader should refer to the appendix to see exactly which *features* are included in each *feature group*.) For our experiments of the next section, we trained five different RL agents, where each agent used one of the five feature sets for its function approximation.

Table 2: Expected regret of various policies

Policy	(n=5, b=15)	(n=8, b=16)	(n=10, b=20)
BR	0.05669	0.07544	**0.07210**
SCL	**0.05413**	**0.07342**	0.07211
RL(set1)	0.05747	0.07830	0.07473
RL(set2)	0.05791	0.07896	0.07390
RL(set3)	0.05555	0.07528	0.07385
RL(set4)	0.05545	0.07464	0.07248
RL(set5)	0.05537	0.07507	0.07280

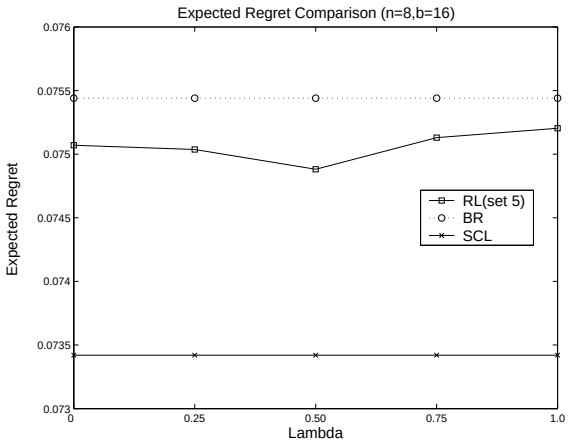

Figure 1: Various values of lambda – BR and SCL still superior to RL

4. EMPIRICAL RESULTS

We conducted experiments on three problems of increasing difficultly, where each initial coin prior was a uniform Beta(1, 1). For each experiment, the expected regret (Equation 2) was calculated for BR, SCL, and the five different policies learned using a TD(0) RL agent. The number of training episodes for the RL agents was set to 1.8 million for the two smaller problems and to 2.8 million for the larger problem. The results are shown in Table 2.

The results indicate that for all problems considered, either BR or SCL produced the smallest expected regret. In fact, no RL policy is able to beat either of the simple policies in the case of ten coins and a budget of twenty, and no RL policy is able to beat SCL on *any* of the problems. We have observed that on larger problems (e.g. ten coins and a budget of thirty), BR beats SCL and RL policies easily. The results of the experiments reveal that despite the extensive number of states observed during training, the RL policies are not generalizing well enough between states to beat the simpler policies.

Additional experiments were run on the $n = 8$, $b = 16$ problem with different values of λ for the TD(λ) learner. For example, Figure 1 shows the results of varying λ when using the fifth set of features for function approximation. For all values of λ considered, the policies learned by RL do only slightly better than BR and are inferior to SCL. The difference between the various TD(λ) learners is not dramatic, but the expected regret is lowest with an intermediate value of $\lambda = 0.5$.

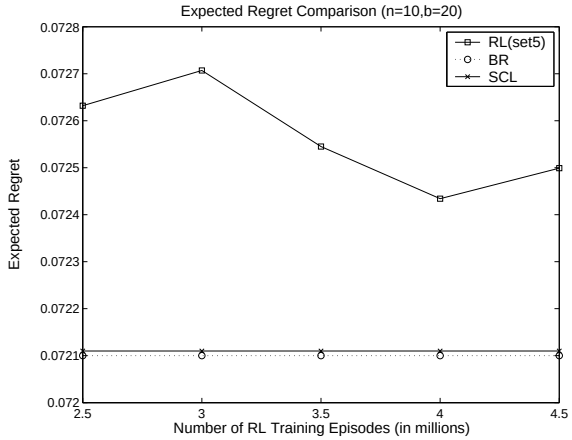

Figure 2: Various amounts of training, RL still exhibits lower performance than simple policies

Table 3: Resources used by each policy on n=10, b=20

Policy	Training time (mins)	Memory Used (MB)
BR	0	0
SCL	0	0
RL(set 5)	630	760

A possible explanation for the lower performance of RL is that not enough training episodes are being experienced. Additional training should permit an RL agent to increase its exploration of the state space, and yield a better policy. To test the effect of increased training, we conducted experiments on the $n = 10, b = 20$ problem in which we varied the number of training episodes from two and a half million up to an even more generous four and a half million. Learning took place with a TD(0.5) learner, using one of the strongest RL feature sets we tested, set number five. The downward sloping trend of Figure 2 suggests that increased training does improve the resulting policy; however, even after four million episodes, the expected regret of the RL policy is still larger than BR and SCL.

For further comparison, we consider the training time and memory required by BR, SCL, and the RL policy after four and half million training episodes. The memory considered is only the policy specific storage (i.e., above and beyond the basic elements such as the beta hyperparameters and the budget that is generally required by all policies). Examining Table 3, we see that even using almost 800 MB of main memory, RL does not gain a significant advantage over the virtually memoryless BR and SCL routines.

As these experiments show, the performance, speed, and low memory requirements make the simper BR and SCL policies preferable to the use of reinforcement learning. Although it should be possible for an RL agent to do better than these heuristic policies, the experimental results indicate that (at least) more cleverly designed features or a better type of function approximator will be required to achieve this.

Perhaps the clearest argument *against* using RL for active model selection (and hence general budgeted learning) is the opportunity cost of conducting the necessary training. That is, although experience is easy to generate, the time and memory used to train RL methods could be equally well spent running a bottom-up dynamic program (as in Section 3.1) that solves for the optimal value of each state. The dynamic program could compute the optimal policy from some select set of states in the same amount of time it takes a reinforcement learning agent to complete training. In effect, the optimal values from this select set of states could be easily combined with the BR or SCL policies to lower their regret even further, and make it yet more difficult for RL methods to compete with these heuristic policies.

Overall, the poor performance demonstrated by RL methods in our experiments suggests that when considering the larger and much more complicated problem of general budgeted learning, the MDP formulation should be avoided, and one should focus on more tractable heuristic policies that have been shown to work effectively [7, 9].

5. RELATED WORK

Active model selection was originally introduced in [10], although several similar problems have been previously studied. The well-known multi-armed bandit problem [14] is concerned with finding the best object among a set, but rewards are typically accrued throughout, without distinguishing training from testing phases. By contrast, active model selection gives no reward until the final coin is selected, and thus more accurately represents the pure training phase of budgeted learning. Strategies from the adversarial bandit formulation [1] could also be adopted for our problem, but the adversarial assumption is unnecessarily strong for our case, and thus less defensive algorithms can usually perform better on active model selection. A more recent bandit-variant, the max k-arm bandit [3], shares our notion of maximizing a *single* reward over the duration of the MDP. However, [3] allows the single reward to occur on any time step, as opposed to strictly at the terminal states.

Duff [5] studied the Bayesian MDP formulation in active model selection, as a Bayes Adaptive Markov Decision Process (BAMDP). That study also considers various RL methods to approximate an optimal policy for BAMDPs, and chooses some of the same types of features for function approximation that we consider in this work. Moreover, the experimental results concur with our findings, as [5] also reports a gap between the reward of the learned RL policies and the optimal policy. Besides RL, another potential strategy for active model selection is online sparse lookahead [19, 8]. Unfortunately, given the size of the state space, we have found that any tractable (truncated) lookahead usually yields a higher regret than the simple BR and SCL policies.

Numerous results from Machine Learning are related to the coins problem, as they too are concerned with the notion of costs at training time. There are too many works to mention here, but some relevant examples include budgeted learning [7, 9], active learning [4], active feature value acquisition [12], and progressive sampling [13]. The field of experimental design [2] is also related, as it deals with how to make finite allocations among objects.

6. CONCLUSIONS AND FUTURE WORK

Despite their known shortcomings, the Biased Robin and Single Coin Lookahead strategies yield low expected regret, and thus an interesting open problem is determining their approximability characteristics. Since features will often have different costs in the general budgeted learning problem, another avenue for future work involves removing the assumption that coins have identical costs, and finding effective strategies in this context. Finally, the fact that the optimal solution can be computed for small versions of the active model selection problem may be useful in deriving an approximation algorithm. One possibility worth exploring is to use careful abstractions to transform large problems into a more tractable, solvable size.

Contributions: This paper investigates the use of reinforcement learning to develop policies to address the active model selection task. We describe deficiencies in the best existing policies, which motivate the need for a more robust solution. We extensively train multiple RL agents using different feature sets for function approximation. Our experiments, on various size problems, demonstrate that the simple policies are able to achieve lower regret with far less computation than the learned RL policies. These results are most helpful when considering approaches for the general problem of budgeted learning. Specifically, in the absence of better features for function approximation, we recommend *not* applying RL techniques to the higher dimensional and more difficult budgeted learning problem, as we anticipate they will prove ineffective.

Acknowledgments

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

7. REFERENCES

[1] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: the adversarial multi-armed bandit problem. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, 1995.

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

[3] V. A. Cicirello and S. F. Smith. The max k-armed bandit: a new model of exploration applied to search heuristic selection. In *AAAI*, 2005.

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

[5] M. Duff. *Optimal learning: computational procedures for Bayes-adaptive Markov Decision Processes.* PhD thesis, University of Massachusetts Amherst, 2002.

[6] D. Hoel and M. Sobel. Comparisons of sequential procedures for selecting the best binomial population. In *Sixth Berkeley Symposium on Mathematical Statistics and Probability*, 1971.

[7] A. Kapoor and R. Greiner. Learning and classifying under hard budgets. In *European Conference on Machine Learning*, 2005.

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

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

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

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

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

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

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

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

[16] R. S. Sutton. Learning to predict by the method of temporal differences. *Machine Learning*, 1988.

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

[18] J. N. Tsitsiklis and B. V. Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 1997.

[19] T. Wang, D. Lizotte, M. Bowling, and D. Schuurmans. Bayesian sparse sampling for on-line reward optimization. In *International Conference on Machine Learning*, 2005.

APPENDIX

A. PROPOSITION ONE

PROOF. Proposition 1. There are numerous ways to prove the proposition; we use a simple subproblem to obtain the result. Consider a state, Q, in which $b' = 1$, there exists two Beta$(3, 2)$ coins, and $(n-2)$ Beta$(2, 2)$ coins. It is easy to verify (using Equation 3) that the optimal action in Q is strictly to flip a Beta$(3, 2)$ coin. To prove the proposition, we show that BR encounters at least K different variants of Q in which it chooses to flip a Beta$(2, 2)$ coin.

Let there be $n = K + 2$ coins, and a budget of $b = 2n + 3$. Notice the budget is such that state Q is guaranteed to occur under BR's strategy. In fact, Q occurs multiple times because there are $\binom{n}{2}$ distinct ways to place the two Beta$(3, 2)$ coins. We also note that since the number of tails on all n coins is equal, we are guaranteed that BR will be currently flipping the first coin in the set. Thus, BR will make a suboptimal decision whenever it reaches state Q with the first coin being one of the Beta$(2, 2)$s. Observe that there are $\binom{n-1}{2}$ distinct versions of state Q in which the first coin is a Beta$(2, 2)$. Now the proposition follows from the fact that: $\binom{n-1}{2} = \frac{(n-1)(n-2)}{2} = \frac{(K+1)K}{2} \geq K$

for all $K \geq 1$. $\square$

B. FEATURE GROUPS

Budget

- remaining budget (b')

Beta Hyper Parameters

- $\alpha_i \quad \forall i = 1..n$
- $\beta_i \quad \forall i = 1..n$

Means and Standard Deviations

- $\mu_i \quad \forall i = 1..n$
- $\sigma_i \quad \forall i = 1..n$

Mean Stats

- $\max_i \mu_i$
- $\min_i \mu_i$
- $\sum_i \mu_i$

Lookahead Stats

- $\max_i \frac{\alpha_i + b'}{\alpha_i + \beta_i + b'}$
- $\frac{\sum_i \left(\frac{\alpha_i + b'}{\alpha_i + \beta_i + b'} \right)}{n}$

Confidence Interval Stats

- $\max_i (\mu_i + 1.96\sigma_i)$ (95% interval)
- $\max_i (\mu_i + 1.28\sigma_i)$ (80% interval)
- $\max_i (\mu_i + 0.67\sigma_i)$ (50% interval)
- $\max_i (\mu_i + 0.126\sigma_i)$ (10% interval)
- $\sum_i (\mu_i + 1.96\sigma_i)$
- $\sum_i (\mu_i + 0.126\sigma_i)$
- $\max_i (\mu_i + b' \times \sigma_i)$
- $\sum_i (\mu_i + b' \times \sigma_i)$