

Reinforcement Learning with the Use of Costly Features

Robby Goetschalckx¹, Scott Sanner², and Kurt Driessens¹

¹ Declarative Languages and Artificial Intelligence, Katholieke Universiteit Leuven, Leuven, Belgium, email: {robby,kurtd}@cs.kuleuven.be

² National ICT Australia, email: Scott.Sanner@nicta.com.au

Abstract. In many practical reinforcement learning problems, the state space is too large to permit an exact representation of the value function, much less the time required to compute it. In such cases, a common solution approach is to compute an approximation of the value function in terms of state features. However, relatively little attention has been paid to the cost of computing these state features. For example, search-based features may be useful for value prediction, but their computational cost must be traded off with their impact on value accuracy. To this end, we introduce a new cost-sensitive sparse linear regression paradigm for value function approximation in reinforcement learning where the learner is able to select only those costly features that are sufficiently informative to justify their computation. We illustrate the learning behavior of our approach using a simple experimental domain that allows us to explore the effects of a range of costs on the cost-performance trade-off.

1 Introduction

We examine cost-sensitive linear-value function approximation in a reinforcement learning context where certain state features are only available at a certain cost. This cost could reflect time or other resources spent on the process of acquiring the feature information, but we assume that this cost can be transformed into the same units used to represent reward in the original reinforcement learning problem.

As a motivating example, consider an agent playing a game of perfect information such as Backgammon or Othello where the opponent executes a stationary policy. The agent knows the rules and thus has access to an accurate model of the environment, except for the opponent policy, which we assume to be unknown. While any reinforcement learning problem of this nature can be solved in theory by using an exact enumerated-state representation of the value function, this is often infeasible in practice due to time and space constraints. Thus, we must often resort to techniques for computing an approximation of the value function in terms of state features.

While value function approximation is well-addressed in the reinforcement learning literature (c.f. Chapter 8 of [1]), the cost of feature computation is often considered negligible and thus ignored. However, continuing our game-playing example, we note that costly search-based state features may be useful for predicting the value of a state. For instance, a useful state feature in a game might be the result of an n -ply expected minimax search. However, there will often be a limit on the time available for a game player to make decisions – either for the entire game or per turn – after which the game

is forfeited. In such a setup, it is important to find a good trade-off between the cost of computation necessary to make a decision and the quality of the resulting decision.

Various theoretical approaches are possible to model this trade-off. While the original optimal reinforcement learning problem we consider can be modelled as a Markov decision process (MDP), one might consider modelling the function approximation setting as a partially observable MDP (POMDP) [2], using information-gathering actions to represent the computation of costly features. In theory, an optimal policy for this POMDP would select those features to compute at any decision stage in order to optimally trade-off feature cost w.r.t. its impact on future reward. However, such a framework requires embedding an already difficult-to-solve MDP inside a POMDP and then solving that POMDP.³ Such an approach will not generally be feasible in practice.

Here we propose a more pragmatic approach where we learn the relative value of features in an explicit way. To do this, we approximate the value function using cost-sensitive sparse linear regression techniques, trading off prediction errors with the costs induced by using a feature. While such an approach does not guarantee that the optimal set of features will be chosen at any decision-stage w.r.t. the cost-performance trade-off, it does guarantee that the prediction accuracy will improve by at least the total cost of the features used.

2 Related Work

Cost-sensitive regression and cost-sensitive classification in non-sequential decision making, supervised learning settings have been widely studied, for example in [3]. While [4] addresses one aspect of cost-sensitive sequential decision-making, it should be noted that in their special case, cost is only associated with actions – not with observing state features – so standard reinforcement learning techniques can be applied without modification. In our work, we specifically consider the case of reinforcement learning with function approximation where state features are costly to compute. This induces a more difficult problem in that standard reinforcement learning techniques must be modified to trade off the cost of using a feature w.r.t. its impact on prediction accuracy during value approximation.

Other work which handles reinforcement learning with costs is discussed in [5]. Here the costly features are considered to have binary values, which allows for the construction of a cost-sensitive regression tree (related to a cost-sensitive classification using a decision tree) where the value of a computed feature can be used to decide which other features to use. In contrast, here we consider the case of linear-value function approximation with real-valued costly features. In the presence of both binary and real-valued features, these two approaches could be merged although such extensions are beyond the scope of the current paper.

The value of information has been formalized by Howard [6] and can be used as a framework to estimate the expected utility increase of observing a random variable (e.g., a feature) given prior information. The meta-reasoning paradigm [7] extends this idea to sequential decision-making by trading off the allocation of computational resources

³ This is only one difficulty with the POMDP approach. See Section 2 for further discussion.

over time w.r.t. the expected gain of using those resources. The difference between the meta-reasoning paradigm and our work is that we do not (and for all practical purposes, cannot) directly model the predictions of costly features since the ability to accurately model them would preclude the need to actually compute them.

When an MDP has costly-observable state (i.e., not state features, but the underlying state itself), the problem may be formally modeled as a POMDP. Variants of such approaches are explored in [8] and [9]. However, such problems are inherently more difficult than the case we consider here. In our case, the underlying problem we are trying to solve is an MDP with fully observable state (zero-cost, by definition), not a POMDP. Our difficulties with costly features only arise when considering the value function approximation paradigm. While we could use a POMDP model to formalize the decision-theoretic trade-off between feature computation and prediction accuracy in this approximation, such an approach would be impractical: Not only would the POMDP be intractably large to solve, without a model of information-gathering actions⁴, the POMDP solution would only become further complicated by the need to perform belief-state updating on a model of such actions from experience. While analytical solutions to related problems exist in theory (c.f., [10]), such approaches are incapable of practically scaling beyond all but the smallest problems.

3 Reinforcement Learning with Costly Features

In this section, we review the general framework of function approximation in reinforcement learning and then proceed to describe our modifications to accommodate costly features.

3.1 MDPs and Reinforcement Learning

We assume the decision-making environment to be a *Markov decision process* (MDP) [11] with which an agent interacts by repeatedly executing an action in the current state, receiving a reward signal and then stochastically transitioning to a successor state. Formally, an MDP can be defined as a tuple $\langle S, A, T, R, \gamma \rangle$. $S = \{s_1, \dots, s_n\}$ is a finite set of fully observable states. $A = \{a_1, \dots, a_m\}$ is a finite set of actions. $T : S \times A \times S \rightarrow [0, 1]$ is a stationary, Markovian transition function. We often express T as the conditional probability distribution $P(s'|s, a)$. We will assume that a reward $R : S \times A \rightarrow \mathbb{R}$ is associated with every state and action. γ is a discount factor s.t. $0 \leq \gamma < 1$ ⁵ used to specify that a reward obtained t timesteps into the future is discounted by γ^t .

A policy $\pi : S \rightarrow A$ specifies the action $a = \pi(s)$ to take in each state s . The value $Q^\pi(s, a)$ of taking an action a in state s and then following the policy π thereafter can

⁴ If an accurate model of a costly information-gathering action existed, it could be substituted in place of the action itself to obtain an equivalent zero-cost action.

⁵ With modifications to enforce that total accumulated reward is finite, $\gamma = 1$ could be accommodated.

be defined using the infinite horizon, expected discounted reward criterion:

$$Q^\pi(s, a) = E_\pi \left[\sum_{t=0}^{\infty} \gamma^t \cdot r_t \mid s_0 = s, a_0 = a \right] \quad (1)$$

where r_t is the reward obtained at time t (assuming s_0 and a_0 respectively represent the state and action at $t = 0$). Then we can define a value function $V^\pi(s) = Q^\pi(s, \pi(s))$ that represents the expected value obtained by starting in state s and acting according to π .

The objective in an MDP is to find a policy π^* such that $\forall \pi, s. V^{\pi^*}(s) \geq V^\pi(s)$. At least one such optimal policy is guaranteed to exist and, in addition, the following Bellman optimality equation is known to hold for π^* [11]:

$$V^{\pi^*}(s) = \max_{a \in A} \left\{ R(s, a) + \gamma \cdot E_{\pi^*} \left[V^{\pi^*}(s_{t+1}) \mid s_t = s \right] \right\}$$

In the *reinforcement learning* setting, the transition and reward model may not be explicitly known to the agent although they both can be sampled from experience. Here, we assume the *generalized policy iteration* (GPI) framework that is known to capture most reinforcement learning approaches [1]. GPI interleaves policy evaluation and policy update stages as follows:

Generalized Policy Iteration (GPI)

1. Start with arbitrary initial policy π_0 and set $i = 0$.
2. Estimate $Q^{\pi_i}(s, a)$ from experience (e.g., using Equation 1).
3. Let $\pi_{i+1}(s) = \arg \max_{a \in A} Q^{\pi_i}(s, a)$.
4. If termination criteria not met, let $i = i + 1$ and goto step 2.

Every reinforcement learning algorithm that is an instance of the above GPI algorithm may prescribe its own method for performing each step and many such instances are known to have strong convergence guarantees. For now, we keep our treatment of reinforcement learning with costly features as general as possible. Specifically, this means that in the context of GPI, we can restrict our discussion of reinforcement learning with costly features to Q-value estimation.

3.2 Cost-sensitive Value Approximation

In practice, it is often infeasible to work with an enumerated state representation due to time and space constraints. A common solution approach in this case is to resort to value function approximation in step 2 of the GPI algorithm by defining relevant state features. In this case, the agent does not directly observe the exact state s of the environment. Instead, the agent has access to a set of state features $F = \{f_1, \dots, f_k\}$, where for each $f \in F$, $f : S \rightarrow \mathbb{R}$ is an (apriori unknown) mapping from a state to \mathbb{R} . The benefits of this approach are well-known: (1) an accurate approximation of a

Q-function (and thus implicitly, a policy) can often be represented with $|F| \ll |S|$ and (2) a limited set of descriptive features enables *generalization* of learned value across multiple states, leading to faster learning.

However, as argued in Section 1 for the games setting, it is plausible to consider using costly features such as those that perform search. Thus, we assume that each feature f is associated with a cost function $c_f : S \rightarrow \mathbb{R}$, which represents the cost of computing feature f in state s .⁶ We assume that the feature cost functions c_f and the reward R are expressed in the same units. For a game where there is a fixed time available per move (after which the game is forfeited), a feature cost could be set to a fraction of the loss value corresponding to the time spent computing the feature.

In the setting of value function approximation with costly features, we must modify our MDP solution criterion to consider both the original reward signal as well as the cost of computing a particular choice of features F . To facilitate this modification, we introduce a new meta-policy $\Pi = \langle \pi, \mathcal{F} \rangle$ where π is the policy for the original MDP and $\mathcal{F} : F \times S \times A \rightarrow \{true, false\}$ is a feature selection function that indicates which features should be selected when evaluating the Q-value for a given state and action. Abusing notation slightly, we often use $\mathcal{F}(s, a)$ to directly denote the subset of features $F' \subseteq F$ selected for a given state s and action a .

However, there is one additional and important complication to value function approximation with costly features. Since we do not represent the policy explicitly, but rather implicitly by evaluating a set of Q-functions, we must take into account the cost of evaluating a set of Q-functions for policy π w.r.t. our feature selection criterion \mathcal{F} . In light of this issue and our previous definitions, we now formally define our problem:

Definition 1 (Cost-sensitive Value Approximation).

Given an MDP $\langle S, A, T, R, \gamma \rangle$, a set of state features F and their related cost functions c_f and a policy π

Find a feature selection function \mathcal{F}^* for meta-policy $\Pi^* = \langle \pi, \mathcal{F}^* \rangle$, such that

$$\mathcal{F}^* = \arg \max_{\mathcal{F}} \left\{ E_{s \sim P(s|\pi, \mathcal{F})} \left[V^{\langle \pi, \mathcal{F} \rangle}(s) \right] \right\} \quad (2)$$

where $P(s|\pi, \mathcal{F})$ is a state occupancy distribution induced for meta-policy $\langle \pi, \mathcal{F} \rangle$ and

$$V^{\langle \pi, \mathcal{F} \rangle}(s) = E_{\langle \pi, \mathcal{F} \rangle} \left[\sum_{t=0}^{\infty} \gamma^t \left(r_t - \sum_{a'_t \in A} \sum_{f \in \mathcal{F}(s_t, a'_t)} c_f(s_t, a'_t) \right) \middle| s_0 = s \right] \quad (3)$$

In words, $V^{\langle \pi, \mathcal{F} \rangle}(s)$ represents the infinite-horizon discounted reward starting from state s and following policy π thereafter. In addition to the reward r_t accumulated at time t , this value definition also explicitly models the cost of computing the meta-policy

⁶ We can easily relax both the feature mapping and its cost function to be stochastic (e.g., for randomized search-based features) since our reinforcement learning approach is sample-based. We provide the deterministic case here to simplify our notation and presentation.

at time t via the cost of computing a Q-function for each action a'_t w.r.t. $\mathcal{F}(s_t, a'_t)$. The objective itself is to find the feature selection function \mathcal{F} that maximizes the value $V^{\langle \pi, \mathcal{F} \rangle}(s)$ for each state s , weighted by the occupancy probability of s w.r.t. the meta-policy.

Perhaps one of the most interesting observations about value function approximation with costly features is that even though π is assumed to be fixed, the actual policy executed varies according to \mathcal{F} . This occurs because in the function approximation setting, the policy is computed implicitly w.r.t. Q-values, which are themselves modulated by a feature selection function \mathcal{F} . In this sense, it appears quite difficult to optimally compute Definition 1 without exhaustive enumeration of all possible \mathcal{F} thus requiring $2^{|\mathcal{F}|}$ evaluations. This is intractable for sufficiently large $|\mathcal{F}|$ and thus we focus on approximate solutions with weaker optimality guarantees in the next section.

4 Sparse Linear-value Approximation

So far we have not assumed a specific functional form for our value approximation. However, from here out, we focus solely on linear value-approximation techniques, not only because linear regression is one of the most widely used and well-understood function approximation methods, but also because it admits elegant sparse solutions that will be useful in minimizing feature usage, and thus feature cost.

We represent a value approximation $\hat{V}_{\mathbf{w}}^{\langle \pi, \mathcal{F} \rangle}(s)$ of $V^{\langle \pi, \mathcal{F} \rangle}(s)$ from Equation 3 as a linear combination of features with weights $\mathbf{w} = \langle w_0, \dots, w_k \rangle$ with each $w_i \in \mathbb{R}$:

$$\hat{V}_{\mathbf{w}}^{\langle \pi, \mathcal{F} \rangle}(s) = w_0 + \sum_{f_i \in \mathcal{F}(s)} w_i f_i(s) \quad (4)$$

Here $\mathcal{F}(s)$ is not considered to be action-dependent and thus we drop the action argument.

When a policy cannot be derived directly from a value function, we could use action-dependent weights w_a for all $a \in A$ to learn a Q-value approximation for each action:

$$\hat{Q}_{\mathbf{w}}^{\langle \pi, \mathcal{F} \rangle}(s, a) = w_{a,0} + \sum_{f_i \in \mathcal{F}(s,a)} w_{a,i} f_i(s) \quad (5)$$

For simplicity, we focus on direct value approximation in the following presentation although the framework can be easily modified to handle Q-value approximation as well.

4.1 Least Angle Regression Methods

Due to their sparsity properties, we focus on a class of linear regression techniques collectively referred to as *least-angle regression* (LAR) methods, such as *lasso* and *forward stagewise regression* [12].

In the context of linear-value approximation, LAR methods provide a solution to the following linear regression problem with cost budget C where we define the indicator

function $\mathbb{I}[\cdot]$ to take the value 1 when its argument \cdot is true and 0 otherwise:

$$\begin{aligned} \text{Minimize: } & \sum_s P(s|\pi, \mathcal{F}) \left[\hat{V}_w^{\langle \pi, \mathcal{F} \rangle}(s) - V^{\langle \pi, \mathcal{F} \rangle}(s) \right]^2 \\ \text{Subject to: } & \left[\sum_s P(s|\pi, \mathcal{F}) \sum_{i=0}^k (\alpha |w_i| + \mathbb{I}[w_i \neq 0] \cdot c_{f_i}(s)) \right] \leq C \end{aligned} \quad (6)$$

Although notationally cumbersome, this optimization problem simply states that we wish to minimize the sum-of-squared errors of the approximate value function $\hat{V}_w^{\langle \pi, \mathcal{F} \rangle}(s)$ w.r.t. samples from the true distribution $V^{\langle \pi, \mathcal{F} \rangle}(s)$ weighted by state occupancy probability. The constraints simply state that the total sum of weights and feature costs for non-zero weights should be less than C . Here, α is a constant indicating how important weight regularization is relative to minimizing the costs. For ordinary LAR, this value is equal to 1 (while the costs are zero). For our preliminary experiments as presented in section 5, we chose $\alpha = 0$.

At first, the budget C would seem to be an unnecessary since the feature costs are already accounted for in the value function estimate. However, including this additional constraint has two advantages: (1) the use of small budgets C encourage sparsity in the weights, thereby maximizing the predictive power of the subset of features with non-zero weights, and (2) by incrementally increasing C from 0 to ∞ , we can greedily add in new features from F , thus providing us with an efficient way to explore the entire feature selection function without enumerating all possibilities.

Fortunately, the forward-stagewise regression solution to optimizing the above problems provides us with an efficient way of doing exactly this. We briefly describe this below and refer the reader to the detailed discussion in [12] for more information on this and related methods. Our adaptation of the forward-stagewise regression algorithm, dubbed FOVEA, can be seen in Fig. 4.1.

Note that we do not normalize the target values, which is the normal procedure for least-angle regression. Our reason for this is that by not dividing by the standard deviation, the residuals at each step correspond to the actual errors. This is necessary for the trade-off of cost and error.

It should be noted that the forward stage-wise approach is a *greedy* selection approach. Because of this, the approximation obtained might not be the optimal one in all cases. A local optimum is guaranteed, however and the increase in value prediction accuracy of using this greedy feature set is guaranteed to equal or exceed the cost of its computation.

5 Experiments

A first setting we used for experiments is a simple deterministic corridor domain (figure 2). The state space consists of five rooms, labeled s_1, \dots, s_5 . From each state two actions, $+1, -1$ can be performed. Performing action $+1$ in state s_i for $i < 5$ leads to s_{i+1} and performing -1 in state s_i for $i > 1$ leads to s_{i-1} . All other actions take the agent to the center s_3 . A reward of 1 is assigned for taking $+1$ in s_5 and -1 is assigned

Forward-stagewise Value Approximation (FOVEA)

1. Normalize all feature predictions in F to have 0 mean and a variance of 1.
2. Initialize the step-size η to some small positive value.
3. Initialize \mathcal{F} such that $\forall s. \mathcal{F}(s) = \emptyset$.
4. Given a policy π and current \mathcal{F} , collect a batch of data samples $V^{\langle \pi, \mathcal{F} \rangle}(s)$ by executing meta-policy $\langle \pi, \mathcal{F} \rangle$ (this implicitly generates samples with state distribution $P(s|\pi, \mathcal{F})$), initialize w_0 with the average value of the batch (this gives the residuals a mean of 0), and repeat the following:
 - (a) calculate the correlation score (absolute correlation with the current residual $r(s)$, *minus the cost* if the feature is not yet included in the linear function) for every feature:

$$score_i = \left| \sum_s f_i(s)r(s) \right| - (1 - \delta_i) \sum_s cost_i(s)$$

(here δ_i indicates whether the feature is already included in the sum: if $f_i \in \mathcal{F}$, $\delta_i = 1$, otherwise $\delta_i = 0$).

- (b) Find the feature f_i with the highest score, and with $score_i \geq 0$, if no such feature found, halt algorithm
- (c) If the f_i was not yet included in \mathcal{F} , let $F = F \cup \{f_i\}$
- (d) Increment or decrement w_i by η to reduce the residuals.

Fig. 1. The FOVEA Algorithm

for taking action -1 in s_1 . All other rewards are equal to 0. We used a discount factor $\gamma = 0.9$.

We provided seven state-action indicator features f_i for $1 \leq i \leq 7$ to the agent where taking action $a \in \{+1, -1\}$ in i results in $f_{i+a} = 1$ with all remaining indicator features set to 0. f_0, f_2, f_3, f_5 and f_6 are free while f_1 and f_4 have a cost c . Furthermore two random number generators were provided to the agent, one which was free and another one which had a high cost 0.5. Finally, the state-action feature indicators f_0, \dots, f_6 were copied but now with the higher cost 0.5. We used FOVEA to approximate Q-values using the state-action features defined above. We used 100 samples for each update unless stated otherwise. All results shown are averages over 10 runs.

A first experiment was performed with $c = 0$. This was an initial check to verify that our algorithm works as expected. Indeed, the agent always learned to use the free, informative features, and never to use the random features or the costly ones.

In a second experiment we varied the value of c over a range of 0 to 0.5. Increasing c takes away the possibility for the agent to locate itself exactly without paying any cost: if the agent does not pay c , it can not distinguish between x_1 and x_4 . (Paying for only f_1 or f_4 is enough, however: if the agent knows he's not in one of the freely observable states, and not in x_4 , he must be in x_1 and vice versa.) We predicted that there is a certain threshold-value for the cost where the agent will be undecided on using one of the features f_1 or f_4 and paying the cost or not using these. If c is much lower, the agent

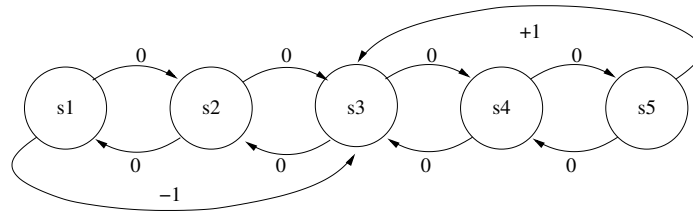


Fig. 2. A simple domain with five linked states. Actions and their corresponding transitions are labeled with their reward.

will always use one of the features, if it is much higher it will not pay off to spend the cost in exchange for the information. For the forward-selection approach, this is exactly what happened with a clear-cut phase transition near $c = 0.185$. This is actually the theoretically correct value for the threshold in this problem domain. This can be clearly seen in figure 3.

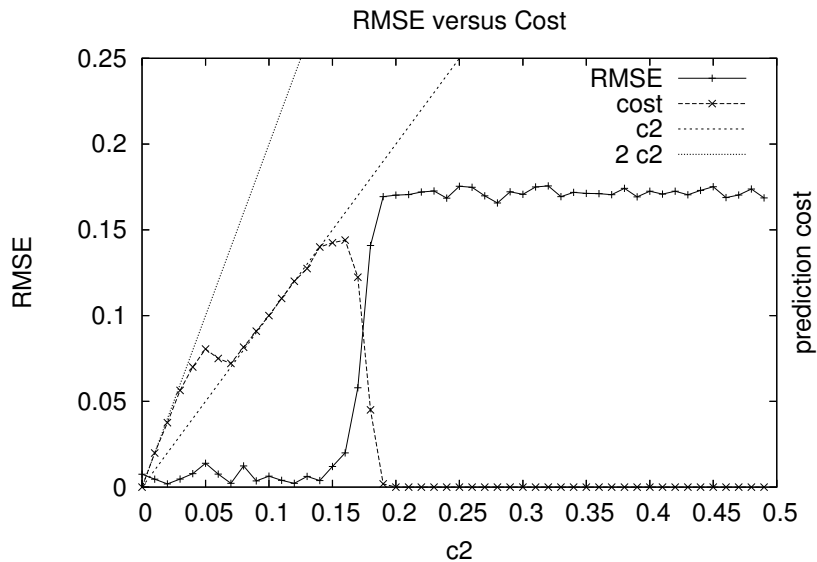


Fig. 3. Error the agent is aware of making versus the amount spent on costly features

Here the cost the agent pays is compared with the prediction error the agent is aware of making (the root mean squared error, or the 2-norm of the final residual). For very low values of c , the agent actually computes both of the features while only one is needed. As the costs are indeed very low, this does not pose a real problem. Using larger batches to compute the new linear regression function would remedy this problem. For

low values of c , the agent keeps paying for one of the features (so the spent cost is equal to c). When getting close to the threshold value of 0.185, however, the agent is no longer willing to pay for the information. Indeed, for values higher than the threshold, the error the agent is aware of making is lower than the cost of extra information. Given longer training time and larger batches for the updates, the phase transition would become even more clear.

For varying values of c the root mean square error of the predictions (compared to the actual optimal values) as the number of examples increases is shown in figure 4. For the value of 0.185 only in about half of the runs the agent was deeming it worth the investment. As this is the threshold value, for which the agent should in theory be undecided whether or not to pay for the extra information, this is what was expected. For lower values than this, the agent quickly learns that the feature is worth its cost. (In figure 4 results are shown for every 1000th episode. During the first 1000 episodes, there is a slight difference for the lower costs, with faster convergence when the information is cheaper.)

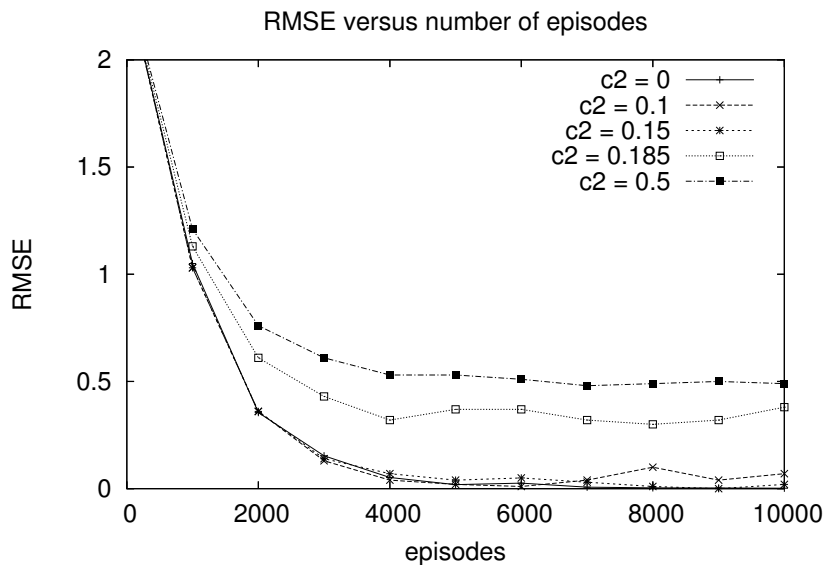


Fig. 4. Evolution of the average prediction error compared to the ideal solution as the number of episodes increases

For a third experiment the size of the domain was increased to see how the algorithm scales up. The domain still consists of a corridor, with a positive and negative reward at respectively the rightmost and the leftmost room. The features provided were state-action indicators as before. Each indicator had a cost inversely proportional to the world size (this compensates for the inherently lower differences in value function for

neighboring states). In figure 5 one can see the sum of the relative errors on the predictions of all state values as the number of episodes increases (note the logarithmic scale on the vertical axis). The value approximation was updated each 10000 samples.

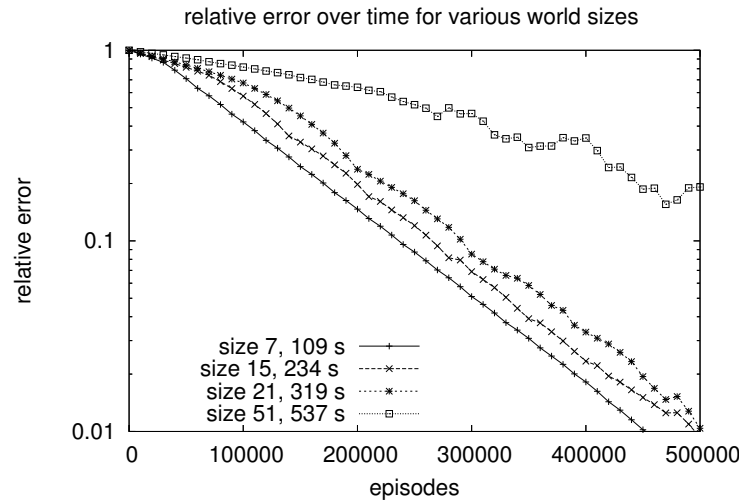


Fig. 5. Relative error on the predictions for varying world sizes

From this figure it is clear that for all these world sizes there was similar convergence behavior. For the larger world (size 51), the convergence is slower, which is understandable, as it takes more time to propagate updates over the entire domain. In the legend of figure 5 the run-times for these domains is also shown. As predicted, the runtime is about linear in the number of features, in this case the size of the world.

6 Conclusions and Future Work

Faced with the task of value approximation in reinforcement learning with costly features, we introduced a novel sparse linear-value approximation approach to efficiently select the features for value prediction that are sufficiently informative to justify their computation. In experimentation, our forward-stagewise value approximation algorithm provided near-perfect trade-offs in value prediction exhibiting sharp phase transitions at theoretical switchover points where feature computation no longer paid-off in reward gain. Furthermore, our experiments demonstrated the ability of our approach to scale in terms of performance and training samples over a range of problem sizes.

While we could only provide an initial investigation of reinforcement learning with costly features in this work, our results warrant future experimentation on larger more difficult problems such as game domains with search-based features. In addition to

such experimental evaluation, various efficiency enhancements can be explored in the forward-stage-wise regression framework to avoid discarding samples every time the feature selection function is updated. Altogether, such advances should make possible a new and useful paradigm for large-scale reinforcement learning in real-world domains where useful features cannot always be assumed to be cost-free.

Acknowledgements

This research was sponsored by the fund for scientific research (FWO) of Flanders, of which Kurt Driessens is a postdoctoral fellow, and by National ICT Australia.

References

1. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. The MIT Press, Cambridge, MA (1998)
2. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. *Artificial Intelligence* **101** (1998) 99–134
3. Domingos, P.: Metacost: A general method for making classifiers cost-sensitive. In: Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining. (1999) 155–164
4. Pednault, E., Abe, N., Zadrozny, B.: Sequential cost-sensitive decision making with reinforcement learning. In: KDD '02: Proceedings of the International Conference on Knowledge discovery and data mining, New York, NY, USA, ACM (2002) 259–268
5. Goetschalckx, R., Driessens, K.: Cost sensitive reinforcement learning. In Kuter, U., Aberdeen, D., Buffet, O., Stone, P., eds.: Proceedings of the workshop on AI Planning and Learning. (2007) 1–5
6. Howard, R.A.: Information value theory. *IEEE Transactions on Systems Science and Cybernetics* **SSC-2** (1966) 22–26
7. Russell, S., Wefald, E.: Principles of metareasoning. *Artificial Intelligence* **49** (1991)
8. Zubek, V.B., Dietterich, T.G.: A POMDP approximation algorithm that anticipates the need to observe. In: Pacific Rim International Conference on Artificial Intelligence. (2000) 521–532
9. Fox, R., Tennenholtz, M.: A reinforcement learning algorithm with polynomial interaction complexity for only-costly-observable mdps. In: AAAI. (2007) 553–558
10. Poupart, P., Vlassis, N., Hoey, J., Regan, K.: An analytic solution to discrete bayesian reinforcement learning. In: ICML '06: Proceedings of the 23rd international conference on Machine learning, New York, NY, USA, ACM (2006) 697–704
11. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley, New York (1994)
12. Efron, B., Hastie, T., Johnstone, I., Tibshirani, R.: Least angle regression. Technical report, Statistics Department, Stanford University (2002)