Non-Parametric Approximate Linear Programming for MDPs

Jason Pazis and Ronald Parr
Department of Computer Science, Duke University
Durham, NC 27708
{jpazis,parr}@cs.duke.edu

Abstract

The Approximate Linear Programming (ALP) approach to value function approximation for MDPs is a *parametric* value function approximation method, in that it represents the value function as a linear combination of features which are chosen *a priori*. Choosing these features can be a difficult challenge in itself. One recent effort, Regularized Approximate Linear Programming (RALP), uses L_1 regularization to address this issue by combining a large initial set of features with a regularization penalty that favors a smooth value function with few non-zero weights. Rather than using smoothness as a backhanded way of addressing the feature selection problem, this paper starts with smoothness and develops a non-parametric approach to ALP that is consistent with the smoothness assumption. We show that this new approach has some favorable practical and analytical properties in comparison to (R)ALP.

1 Introduction and motivation

Linear value function approximation is a standard technique for discovering approximate solutions to large Markov Decision Processes (MDPs). One of the major difficulties in linear value function approximation for MDPs is finding a good set of features. If the feature set is too restrictive, it will fail to capture important structure in the value function (bias). On the other hand if the feature set is too expressive, the value function can overfit the training samples (variance). Regularization and feature selection have recently been the focus of much research in the field of reinforcement learning as a potential solution to these problems (Kolter and Ng 2009; Petrik et al. 2010).

Linear value function approximation represents the value of every state with a linear combination of a (possibly nonlinear) set of features. One family of approaches to the feature selection problem starts with an initially large set of features and adds an L_1-norm penalty or constraint on the weights. For example, in the Approximate Linear Programming (ALP) approach for MDPs, which is most closely related to the topic of this paper, the regularized extension (RALP) imposes a hard bound on the L_1-norm of the weights, but solves an otherwise identical linear program.

A penalty or constraint on the weights of a linear value function approximation (regardless of norm) can be viewed as a form of smoothness assumption. When L_1 regularization, which favors solutions with few non-zero weights, is used as a feature selection method, any smoothing effect from the penalty/constraint on the weights is often seen merely as a means to achieving the primary goal of a sparse set of features.

This paper takes a novel approach to ALP that starts with a smoothness assumption on the value function. Using nothing more than the smoothness assumption, we develop an LP approach that is entirely non-parametric, bypassing the need for features and feature selection. The new method is straightforward in implementation, has more easily computed error bounds than ALP, is amenable to problems with large (multidimensional) or even infinite (continuous) action spaces, and (unlike ALP) does not require a model to select actions using the resulting approximate solution.

2 Background

A *Markov Decision Process* (MDP) is a 5-tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is the state space of the process, $\mathcal{A}$ is the action space, P is a Markovian transition model $\big(P(s'|s,a)$ denotes the probability of a transition to state s' when taking action a in state $s\big)$, R is a reward function $\big(R(s,a)$ is the expected reward for taking action a in state $s\big)$, and $\gamma \in (0,1)$ is a discount factor for future rewards. A *deterministic policy* π for an MDP is a mapping $\pi : \mathcal{S} \mapsto \mathcal{A}$ from states to actions; $\pi(s)$ denotes the action choice in state s.

The value $V^\pi(s)$ of a state s under a policy π is defined as the expected, total, discounted reward when the process begins in state s and all decisions are made according to policy π. The goal of the decision maker is to find an optimal policy π^* for choosing actions, which yields the optimal value function $V^*(s)$, defined recursively via the Bellman optimality equation:

$$V^*(s) = \max_a R(s,a) + \gamma \sum_{s'} P(s'|s,a) V^*(s')$$

The value $Q^\pi(s,a)$ of a state-action pair (s,a) under a policy π is defined as the expected, total, discounted reward when the process begins in state s, action a is taken at the

first step, and all decisions thereafter are made according to policy π. Once again our goal is to find an optimal policy π^* for choosing actions, which yields the optimal value function $Q^*(s,a)$:

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q^*(s',a')$$

Reinforcement learning

In reinforcement learning, a learner interacts with a stochastic process modeled as an MDP and typically observes the state and immediate reward at every step; however, the transition model P and the reward function R are not accessible. The goal is to learn an optimal policy using the experience collected through interaction with the process. At each step of interaction, the learner observes the current state s, chooses an action a, and observes the resulting next state s' and the reward received r, essentially sampling the transition model and the reward function of the process. Thus experience comes in the form of (s,a,r,s') samples.

Solving MDPs via linear programming

The exact case One way to solve for the optimal value function V^* is via linear programming, where every state $s \in \mathcal{S}$ is a variable and the objective is to minimize the sum of the states' values under the constraints that the value of each state must be greater than or equal to all Q-values for that state:

$$\begin{aligned} &minimize \sum_s V^*(s) \\ &subject\ to: \\ &(\forall s,a) V^*(s) \geq R(s,a) + \gamma \sum_{s'} P(s'|s,a) V^*(s') \end{aligned}$$

Extracting the policy is fairly easy (at least conceptually), just by picking the action with a corresponding non-zero dual variable for the state in question (equivalently, picking the action that corresponds to the constraint that has no slack in the current state). Note that we can have a set of state-relevance weights $\rho(s)$ associated with every state in the optimization criterion; however, for the exact case every set of positive weights leads to the same V^*.

Approximate linear programming (ALP) In many real world applications the number of states is too large (or even infinite if the state space is continuous), rendering exact representation intractable. In those cases we approximate the value function via a linear combination of (possibly non-linear) basis functions or features. The variables in the approximate linear program are now the weights assigned to each basis function and the value of each state is computed as $\phi(s)^T w$, where $\phi(s)$ is the feature vector for that state, and w is the weight vector. The linear program becomes:

$$\begin{aligned} &minimize && \sum_s \rho(s) \phi^T(s) w \\ &subject\ to: \\ &(\forall s,a)\ \phi^T(s) w &\geq\ & R(s,a) + \gamma \sum_{s'} P(s'|s,a) \phi^T(s') w \end{aligned}$$

Using features dramatically reduces the number of variables in the program, however it does not reduce the number of constraints. Since the number of constraints is larger than the number of variables in the exact linear program, we have to find a way to reduce the number of constraints by sampling. Making certain assumptions over our sampling distribution (de Farias and Van Roy 2004), or if we incorporate regularization (Petrik et al. 2010), we can sample constraints and bound the probability that we will violate a non-sampled constraint, or bound the performance degradation that will occur as a result of missing constraints.

Unfortunately this approximate formulation does not allow for easy extraction of a policy from the dual. Not only is the number of dual variables large (the same as the number of samples) but it does not offer a straightforward way to generalize to unseen states. Choosing an action using a (R)ALP solution typically requires a model to compute Q-values given the approximate value function returned by the linear program. Also note that the state-relevance weights now influence the solution, imposing a trade-off in the quality of the approximation across different states (de Farias and Van Roy 2003).

An alternative way of expressing the ALP (consistent with the notation in Petrik et al. 2010) that also emphasizes the similarity to the exact LP is to express the problem as an optimization within a constrained family of value functions:

$$\begin{aligned} &minimize \sum_s \rho(s) \tilde{V}(s) \\ &subject\ to: \\ &(\forall s,a) \tilde{V}(s) \geq R(s,a) + \gamma \sum_{s'} P(s'|s,a) \tilde{V}(s') \\ &\tilde{V} \in \mathcal{M} \end{aligned}$$

This representation is more general than the typical ALP formulation in that it allows arbitrary constraints on $\tilde{V}$ via $\mathcal{M}$. For the standard ALP approach, $\mathcal{M} = span(\Phi)$, but any specification of $\mathcal{M}$ that can be implemented through linear constraints can be seen as a form of ALP.

3 Related Work

Regularized approximate linear programming (RALP) (Petrik et al. 2010) can be considered as a close parallel of this work for parametric ALPs. Using L_1-norm regularization on the feature weights, RALP is able to select a good set of features from a potentially very large pool, while our work requires no features and only a smoothness constraint on the value function. Another important difference is that our approach is compatible with infinite action spaces and does not require a model for action selection.

Other non-parametric approaches to reinforcement learning include variable resolution grids (Munos and Moore 2002), the work of Ormoneit and Sen (2002), Fitted Q-Iteration with tree-based approximators (Ernst, Geurts, and Wehenkel 2005) and many kernelized methods (Taylor and Parr 2009). Compared to the non-parametric ALP, these methods may require storing significantly more information, may be unable to handle very large action spaces and

characterizing their performance in terms of easily measured model parameters can be a challenge.

4 Non-parametric ALP

If the optimal value function is Lipschitz continuous it satisfies the following constraint for every two states s and s':

$$|V^*(s) - V^*(s')| \leq L_{V^*} d(s, s')$$

where:

$$d(s, s') = ||k(s) - k(s')|| \tag{1}$$

and $k(s)$ is a mapping from the state space to some normed vector space.

We will use the notation $\mathcal{M}_L$ to denote the set of functions with Lipschitz constant L. Observe that for any L, the constraint $\tilde{V} \in \mathcal{M}_L$ can be enforced via linear constraints:

$$(\forall s, s')\tilde{V}(s) \quad \geq \quad \tilde{V}(s') - L_{\tilde{V}} d(s, s') \tag{2}$$

If $L_{\tilde{V}} = L_{V^*}$ and all constraints are present in the LP, the smoothness constraints will have no effect on the solution and $\tilde{V} = V^*$. In practice $L_{\tilde{V}}$ will typically differ from L_{V^*}, either by design (to avoid overfitting) or due to lack of sufficient knowledge of L_{V^*}. Also, it will be impractical to have one constraint per state. Using $\tilde{S}$ to represent the set of state-action pairs for which the Bellman equation is enforced, the non-parametric approximate LP will be:

$$\begin{aligned}
&minimize \sum_s \tilde{V}(s) \\
&subject\ to: \\
&(\forall s \in \tilde{S}, a)\tilde{V}(s) \geq R(s,a) + \gamma \sum_{s'} P(s'|s,a)\tilde{V}(s') \\
&\tilde{V} \in \mathcal{M}_{L_{\tilde{V}}}
\end{aligned}$$

When $\tilde{V} \in \mathcal{M}_{L_{\tilde{V}}}$ is implemented via the linear constraints in 2, this provides a complete specification of NP-ALP. A few subtleties in the LP above bear note: First, the smoothness constraint on $\tilde{V}$ is defined over the entire state space, not just the states in $\tilde{S}$. It might seem that enforcing this constraint would require constraints for every state in S, sacrificing any efficiency gained by sampling the constraints for the Bellman operator. In practice, however, it suffices to implement smoothness constraints only for states in $\tilde{S}$ or reachable in one step from a state in $\tilde{S}$. Smoothness constraints on other states will not influence the solution to the LP. Second, the sum over all states in the objective function may seem impractical. However, summing over the values of states not in $\tilde{S}$ has no effect on the LP solution.

In comparison to standard ALP, the non-parametric ALP has some desirable properties. In standard ALP, the state relevance weights ρ impose a trade-off in approximation resources devoted to different states. While this adds flexibility, it's quite difficult to know what a good value for ρ is in practice (without knowing the stationary distribution of the optimal policy) and ρ becomes yet another difficult to tune input into the function approximation process. In contrast, there is no change in the non-parametric solution if different states are assigned different weights in the objective function. (Since the approximator is non-parametric, there is no sense in which accuracy in one state could be traded off against accuracy in another via ρ.) Another desirable property of the non-parametric ALP is that it ensures a bounded solution even when constraints are sampled. In ordinary ALP, a single missing constraint can, in the worst case, cause the LP to be unbounded.

Using $\tilde{V}$

When presented with an unknown state t that was not directly constrained in the LP, an estimate of its value can be obtained by identifying a constrained state s (variable $V(s)$) that maximizes the bound[1] $\tilde{V}(t) \geq \tilde{V}(s) - L_{\tilde{V}} d(s, t)$. This procedure determines the value that state t would have been assigned by the LP if a smoothness constraint on t had been included. Note that if such a constraint had been included, it would not have changed the value of any other state.

We will call all variables corresponding to state-action pairs for which the corresponding Bellman constraint holds with equality (the dual variables are non-zero) *basic* and the rest *non-basic*. Non-basic variables can be discarded without changing the solution. This is useful both for sparsifying the solution to make evaluation faster and can be used to construct a homotopy method for solving the linear program efficiently.

To see why the above is true, consider a state-action pair s, a corresponding to a non-basic variable. This implies $\tilde{V}(s) = \tilde{V}(s') - L_{\tilde{V}} d(s, s')$ for some state[2] s'. When presented with state s'' to evaluate, we have:

$$\tilde{V}(s'') \quad \geq \quad \tilde{V}(s) - L_{\tilde{V}} d(s'', s) \tag{3}$$

$$\tilde{V}(s'') \quad \geq \quad \tilde{V}(s') - L_{\tilde{V}} d(s'', s') \tag{4}$$

Substituting $\tilde{V}(s) = \tilde{V}(s') - L_{\tilde{V}} d(s, s')$ into 3:

$$\tilde{V}(s'') \quad \geq \quad \tilde{V}(s') - L_{\tilde{V}}(d(s'', s) + d(s, s')) \tag{5}$$

By the triangle inequality $d(s'', s') \leq d(s'', s) + d(s, s')$ thus constraint 3 does not influence the value of $\tilde{V}(s'')$.

Action selection

In the exact LP formulation, the maximizing action is the one corresponding to the non-zero dual variable for that state. For the non-parametric ALP, after non-basic variables have been discarded, there is only one such variable per basic state. For any basic state s, $\tilde{V}(s)$ is bound by a Bellman constraint from state-action pair s, a, so $\tilde{V}(s) = \tilde{Q}(s, a)$. If s bounds the value of a non-basic state t by[3] $\tilde{V}(t) \geq \tilde{V}(s) - L_{\tilde{V}} d(s, t)$, it also implicitly bounds $\tilde{Q}(t, a)$. The predicted optimal action at t will therefore be the same as in s

[1] Note that this operation is quite trivial and does not require solving a linear program.

[2] In the case where $s = s'$ this means that some other action dominates the state-action pair in question.

[3] For simplicity of exposition, we assume that $L_{Q_a} = L_V \ \forall a \in A$. The case where different actions have different Lipschitz constants extends naturally.

since the bounds from other states are lower, implying lower estimated Q-values.

The above has two important consequences. The first is that only actions present in the training set can ever be selected during policy execution, since the value estimation and action selection mechanisms are inherently pessimistic. The second is that action selection complexity is independent of the number of actions. Thus we can deal with spaces with infinite (continuous) or massive (multidimensional) action spaces. Sampling is of course important; however, this goes beyond the scope of this paper (see future work for a brief discussion).

Solving the NP-ALP

One important concern with many non-parametric methods is computation time. For NP-ALP, the cause of concern is the quadratic number of constraints relative to the number of samples, arising from the Lipschitz continuity constraints. In our experiments, this was not a problem; all programs took at most a few minutes to complete on a modern CPU when implemented exactly as specified in section 4.

For larger problems, the size of the NP-ALP could become prohibitive. Fortunately the NP-ALP constraints have favorable properties. All the continuity constraints involve exactly two variables, resulting in a very sparse constraint matrix, a property that modern solvers can exploit. Additionally, for every sample either its Bellman constraint or exactly one of its Lipschitz constraints will be active. The above facts can be used to construct a homotopy method. Starting from $L_{\tilde{V}} = 0$ only one Bellman constraint will be active (the one with the largest reward) and all other states will be bound by Lipschitz constraints to $\tilde{V} = \frac{R_{\max}}{1-\gamma}$. Progressively relaxing $L_{\tilde{V}}$, the entire space of solutions can be traversed. This was not implemented for our experiments.

5 Error bounds

Lemma 5.1. *If V is a representable value function, $V + \epsilon\mathbf{1}$ is also representable, where $\mathbf{1}$ is a vector if all ones.*

It is easy to see that by adding a constant ϵ to all the basic variables, the entire value function is shifted by ϵ.

Definition 5.2. *The set of ϵ-transitive feasible value functions is defined for $\epsilon \geq 0$ as: $K(\epsilon) = V \in R^{|S|} : V \geq TV - \epsilon\mathbf{1}$. A value function is transitive-feasible when $V \geq TV$ ($K = K(0)$).*

Definition 5.3. *$\epsilon_p(L_{\tilde{V}}) \leq d_{\max}(L_{V^*} + L_{\tilde{V}})$ bounds the violation of Bellman constraints missing in the sampled LP, where $d_{\max}$ is the maximum distance from a non-sampled state to the closest sampled state.*

Given lemma 5.1 and the definitions above, the following theorem is a straightforward adaptation of an analogous result for RALP from Petrik et al. 2010.

Theorem 5.4. *Define ϵ to be the closest value function in $\mathcal{M}(L_{\tilde{V}})$ to V^*, $\epsilon = \frac{2}{1-\gamma}\min_{V \in M(L_{\tilde{V}})} ||V - V^*||_\infty$, and let $\tilde{V}$ be the solution to the non-parametric ALP then,*

$$||\tilde{V} - V^*||_1 \quad \leq \quad \epsilon + 2\frac{\epsilon_p(L_{\tilde{V}})}{1-\gamma}.$$

As with RALP, if sampled transitions are used, an additional additive term: $3\frac{\epsilon_s(L_{\tilde{V}})}{1-\gamma}$ would be included, where ϵ_s is derived from the error introduced by using sampled transitions instead of the full Bellman equation. Since the analysis is identical to that of Petrik et al., we omit further discussion of ϵ_s for brevity. The RALP analysis also includes an ϵ_c term which results from discrepancies between the objective function of the original problem and the objective function in the constraint sampling case. This term does not appear in our analysis because the solution is invariant to changes in state relevance weights (see Section 4). A key difference between our bound and that of RALP, is it is easy to connect ϵ to properties of the model.

Theorem 5.5.

$$\epsilon \leq \frac{R_{\max} - R_{\min}}{(1-\gamma)^2}\left(1 - \frac{L_{\tilde{V}}}{L_{V^*}}\right)$$

Proof. If $L_{\tilde{V}} \geq L_{V^*}$, $V^* \in M_{L_{\tilde{V}}}$ and $\epsilon = 0$. Otherwise, the value function V' produced by translating V^* so that $|V^*_{\max}| = |V^*_{\min}|$, scaling by $\frac{L_{\tilde{V}}}{L_{V^*}}$ and translating back to where V^* was, is in $M_{L_{\tilde{V}}}$:

$$V' = \left(V^* - \frac{V^*_{\max} + V^*_{\min}}{2}\right)\frac{L_{\tilde{V}}}{L_{V^*}} + \frac{V^*_{\max} + V^*_{\min}}{2} \in M_{L_{\tilde{V}}}$$

V's distance from V^* is:

$$\begin{aligned} ||V' - V^*||_\infty &= \frac{V^*_{\max} - V^*_{\min}}{2}\left(1 - \frac{L_{\tilde{V}}}{L_{V^*}}\right) \\ &\leq \frac{R_{\max} - R_{\min}}{2(1-\gamma)}\left(1 - \frac{L_{\tilde{V}}}{L_{V^*}}\right) \end{aligned}$$

□

The ratio $\frac{L_{\tilde{V}}}{L_{V^*}}$ may be difficult to determine, but can be bound by more easily determined quantities.

Lemma 5.6. *If the fastest changing state-action value function Q^*_a is $L_{Q^*_{\max}}$-Lipschitz continuous, $L_{V^*} \leq L_{Q^*_{\max}}$.*

Definition 5.7. *If the reward function is L_r-Lipschitz continuous, it satisfies the following constraint for every two states s_1 and s_2:*

$$|r(s_1, a) - r(s_2, a)| \leq L_r d(s_1, s_2)$$

Definition 5.8. *If the transition model is L_p-Lipschitz continuous it satisfies the following constraint for every two states s_1 and s_2, and all V with $L_V = 1$:*

$$\Big|\int_{s'}(p(s'|s_1, a) - p(s'|s_2, a))V(s')ds'\Big| \leq L_p d(s_1, s_2)$$

Observe that this bounds the difference in expected next state values with respect to a normalized V. If $L_V \neq 1$, the worst case difference can be scaled appropriately.

Theorem 5.9. *If $\gamma L_p < 1$:*

$$L_{V^*} \leq \frac{L_r}{1 - \gamma L_p}$$

Proof. For every two states s_1 and s_2:

$$\begin{aligned} L_{Q^*_{\max}} d(s_1, s_2) &\leq |Q^*(s_1, a) - Q^*(s_2, a)| \\ L_{Q^*_{\max}} d(s_1, s_2) &\leq |R(s_1, a) + \gamma \int_{s'} p(s'|s_1, a) V^*(s') \\ &\quad - R(s_2, a) - \gamma \int_{s'} p(s'|s_2, a) V^*(s')| \\ L_{Q^*_{\max}} d(s_1, s_2) &\leq (L_r + \gamma L_p L_{V^*}) d(s_1, s_2) \\ L_{Q^*_{\max}} &\leq L_r + \gamma L_p L_{V^*}. \end{aligned}$$

In the penultimate step above, we have used the definition of L_p and scaled by L_{V^*}. Using lemma 5.6:

$$\begin{aligned} L_{V^*} &\leq L_r + \gamma L_p L_{V^*} \\ L_{V^*} &\leq \frac{L_r}{1 - \gamma L_p} \end{aligned}$$

□

$\gamma L_p < 1$ is satisfied is many noise models, e.g., actions that add a constant impulse with Gaussian noise.

6 Experimental Results

In this section we test the non-parametric approach to approximate linear programming on three popular domains in reinforcement learning and compare its performance against RALP[4]. Given that RALP uses the model to evaluate every action during policy execution, this is an unfair comparison. Nevertheless, the non-parametric approach is able to achieve comparable performance even without the use of a model.

We should also note that all our samples come in the form of (s, a, r, s') samples, thus we only have one constraint per sampled state, which amounts to sampling the right hand side of the Bellman equation[5].

Car on the hill

The car on the hill problem (Ernst, Geurts, and Wehenkel 2005) involves driving an underpowered car stopped at the bottom of a valley between two hills, to the top of the steep hill on the right. The 2-dimensional continuous state space (p, v) includes the current position p and the current velocity v. The controller's task is to (indirectly) control the acceleration using a thrust action $u \in \{-4, 4\}$, under the constraints $p > -1$ and $|v| \leq 3$ in order to reach $p = 1$. The task requires temporarily driving away from the goal in order to gain momentum. The agent receives a reward of -1, if a constraint is violated, a reward of $+1$, if the goal is reached, and a zero reward otherwise. The discount factor of the process was set to 0.98 and the control interval to 100ms.

For both algorithms we scaled the speed portion of the state, from $[-3, 3]$ to $[-1, 1]$. The distance function for the non-parametric approach was chosen to be the two norm of the difference between states and the Lipschitz constant was set to $L_{\tilde{V}} = 1.0$. The features for RALP comprised of a grid of 20×20 equally spaced Gaussian radial basis functions in $[-1.0, 1.0] \times [-1.0, 1.0]$ with $\sigma = 1$ and the regularization parameter was set to $\psi = 100$.

Figure 1 (a) shows the total discounted reward as a function of the number of training episodes. Training samples were collected in advance from "random episodes", that is, starting the car in a randomly perturbed state and following a purely random policy. Each episode was allowed to run for a maximum of 200 steps or until a terminal state was reached, both during sampling and policy execution.

As we can see, even for such a naive choice of distance function and without the use of a model, the non-parametric ALP is able to perform as well as RALP, even surpassing its performance when the number of episodes is limited.

Inverted Pendulum

The inverted pendulum problem (Wang, Tanaka, and Griffin 1996) requires balancing a pendulum of unknown length and mass at the upright position by applying forces to the cart to which it is attached. The 2-dimensional continuous state space includes the vertical angle θ and the angular velocity $\dot{\theta}$ of the pendulum. The *continuous* action space of the process is the range of forces in $[-50N, 50N]$.

Most researchers in reinforcement learning choose to approach this domain as an avoidance task, with zero reward as long as the pendulum is above the horizontal configuration and a negative reward when the controller fails. Instead we chose to approach the problem as a regulation task, where we are not only interested in keeping the pendulum upright, but we want to do so while minimizing the amount of force we are using. Thus a reward of $1 - (u/50)^2$, was given as long as $|\theta| \leq \pi/2$, and a reward of 0 as soon as $|\theta| > \pi/2$, which also signals the termination of the episode. The discount factor of the process was set to 0.98 and the control interval to 100ms. In this setting, making full use of the available action range is required to get good performance.

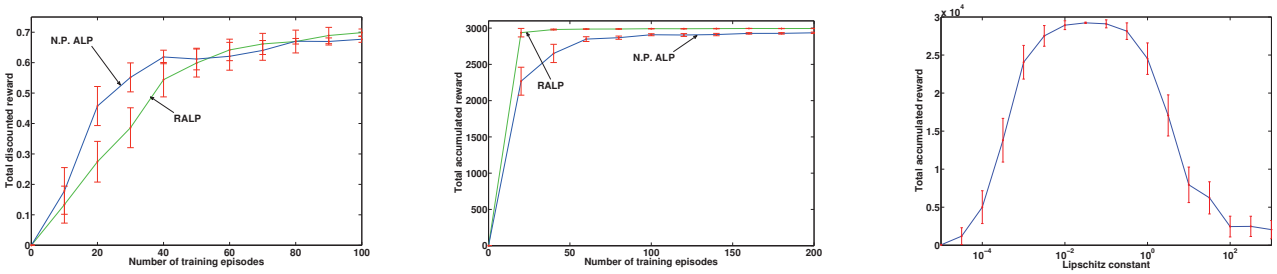
For both algorithms we standardized the state space and used PCA, keeping only the first principal component. The distance function for the non-parametric ALP was chosen to be the two norm of the difference between states and the Lipschitz constant was set to $L_{\tilde{V}} = 10.0$. For RALP the action space was discretized to 256 actions to approximate the continuous range $[-50N, 50N]$, the features where a grid of 200 equally spaced Gaussian radial basis functions in $[-1.5, 1.5]$ with $\sigma = 1$, and the regularization parameter was set to $\psi = 100$.

Figure 1 (b) shows the total accumulated reward as a function of the number of training episodes. Training samples were collected in advance by starting the pendulum in a randomly perturbed state close to the equilibrium state $(0, 0)$ and following a purely random policy.

It is instructive to see why RALP performs (slightly) better in this domain. RALP has access to the full reward and transition model and is able to try every available action before taking it. On the other hand, even though the non-parametric ALP has access to every action in the continuous range, it will only choose actions for which it has samples close enough to the state in question to be dominant. This highlights the importance of sampling.

[4]The Lipschitz constants $L_{\tilde{V}}$ and the regularization parameters ψ for RALP were chosen via cross-validation.

[5]All the domains tested in this paper are noiseless, thus there is only one possible next state for every state-action combination.



(a) (b) (c)

Figure 1: (a), (b): Performance versus training episodes for the car on the hill and inverted pendulum tasks. (c): Performance versus the Lipschitz constant for bicycle balancing. Graphs show averages and 95% confidence intervals over 100 independent runs.

Bicycle Balancing

The bicycle balancing problem (Ernst, Geurts, and Wehenkel 2005), has four state variables (angle θ and angular velocity $\dot{\theta}$ of the handlebar as well as angle ω and angular velocity $\dot{\omega}$ of the bicycle relative to the ground). The *continuous* action space is two dimensional and consists of the torque applied to the handlebar $\tau \in [-2, +2]$ and the displacement of the rider $d \in [-0.02, +0.02]$. The goal is to prevent the bicycle from falling.

Again, we approached the problem as a regulation task, rewarding the controller for keeping the bicycle as close to the upright position as possible. A reward of $1-|\omega|\times(\pi/15)$, was given, as long as $|\omega| \leq \pi/15$, and a reward of 0, as soon as $|\omega| > \pi/15$, which also signals the termination of the episode. The discount factor was 0.98, the control interval was 10ms and training trajectories were truncated after 20 steps. We standardized the state space and used PCA, keeping only the first principal component. The distance function was the two norm of the difference between states.

Figure 1 (c) shows total accumulated reward versus the Lipschitz constant for 100 training episodes. As we can see, the controllers learned have excellent performance for a wide range of Lipschitz constants (notice the logarithmic scale for the x axis).

7 Discussion and future work

This paper introduced a non-parametric approach to ALP which bypasses the need for features, is compatible with continuous actions, and requires nothing more than a smoothness assumption on some function of the state.

As shown, the non-parametric ALP approach does not need a model during policy execution. Furthermore, the amount of information that needs to be stored can be quite small. This is because only state-action pairs corresponding to tight Bellman constraints need to be stored. This allows non-parametric ALP to handle large action spaces well.

The majority of domains used in reinforcement learning experiments, have a very small number of discrete actions. In more realistic domains, sufficient sampling of each action is a challenging problem. In an online learning setting, an approach similar to the one used in this paper to provide pessimistic estimates of states' values could provide optimistic estimates, identifying promising actions for exploration.

One of the biggest strengths of non-parametric methods is the great flexibility in selecting the distance function. However this is simultaneously their greatest weakness. A poor choice can have catastrophic effects to performance. In our experiments we used very simple distance functions. A more sophisticated choice for function $k(s)$ in equation 1 should allow us to tackle much more challenging domains. Apart from statically choosing a distance function from a fixed set, we could also learn one. Recent work in manifold learning is especially relevant.

Acknowlegments

We thank the reviewers for helpful comments and suggestions. This work was supported by NSF IIS-0713435. Opinions, findings, conclusions or recommendations herein are those of the authors and not necessarily those of NSF.

References

de Farias, D. P., and Van Roy, B. 2003. The linear programming approach to approximate dynamic programming. *Operations Research* 51(6):850–865.

de Farias, D. P., and Van Roy, B. 2004. On constraint sampling in the linear programming approach to approximate dynamic programming. *Mathematics of OR* 29(3):462–478.

Ernst, D.; Geurts, P.; and Wehenkel, L. 2005. Tree-based batch mode reinforcement learning. *JMLR* 6:503–556.

Kolter, J. Z., and Ng, A. Y. 2009. Regularization and feature selection in least-squares temporal difference learning. In *ICML 09*, 521–528. ACM.

Munos, R., and Moore, A. 2002. Variable resolution discretization in optimal control. *Machine learning* 49(2):291–323.

Ormoneit, D., and Sen, Ś. 2002. Kernel-based reinforcement learning. *Machine Learning* 49(2):161–178.

Petrik, M.; Taylor, G.; Parr, R.; and Zilberstein, S. 2010. Feature selection using regularization in approximate linear programs for Markov decision processes. In *ICML-10*, 871–878. Haifa, Israel: Omnipress.

Taylor, G., and Parr, R. 2009. Kernelized value function approximation for reinforcement learning. In *ICML 09*, 1017–1024. ACM.

Wang, H.; Tanaka, K.; and Griffin, M. 1996. An approach to fuzzy control of nonlinear systems: Stability and design issues. *IEEE Trans. on Fuzzy Systems* 4(1):14–23.