Regularization and Feature Selection in Least-Squares Temporal Difference Learning

J. Zico Kolter KOLTER@CS.STANFORD.EDU
Andrew Y. Ng ANG@CS.STANFORD.EDU
Computer Science Department, Stanford University, CA 94305

Abstract

We consider the task of reinforcement learning with linear value function approximation. Temporal difference algorithms, and in particular the Least-Squares Temporal Difference (LSTD) algorithm, provide a method for learning the parameters of the value function, but when the number of features is large this algorithm can over-fit to the data and is computationally expensive. In this paper, we propose a regularization framework for the LSTD algorithm that overcomes these difficulties. In particular, we focus on the case of l_1 regularization, which is robust to irrelevant features and also serves as a method for feature selection. Although the l_1 regularized LSTD solution cannot be expressed as a convex optimization problem, we present an algorithm similar to the Least Angle Regression (LARS) algorithm that can efficiently compute the optimal solution. Finally, we demonstrate the performance of the algorithm experimentally.

1. Introduction

We consider the task of reinforcement learning (RL) in large or infinite state spaces. In such domains it is not feasible to represent the value function explicitly, and instead a common strategy is to employ function approximation to represent the value function using some parametrized class of functions. In particular, we consider *linear value function approximation*, where the value function is represented as a linear combination of some set of basis functions. For this task, the Temporal Difference (TD) family of algorithms (Sutton, 1988), and more specifically the Least-Squares TD (LSTD) algorithms (Bradtke & Barto, 1996; Boyan, 2002; Lagoudakis & Parr, 2003), provide a method for learning the value function using only trajectories generated by the system. However, when the number of features is large compared to the number of training samples, these methods are prone to over-fitting and are computationally expensive.

In this paper we propose a regularization framework for the LSTD family of algorithms that allows us to avoid these problems. We specifically focus on the case of l_1 regularization, which results in sparse solutions and therefore serves as a method for feature selection in value function approximation. Our framework differs from typical applications of l_1 regularization in that solving the l_1 regularized LSTD problem cannot be formulated as a convex optimization problem; despite this, we show that a procedure similar to the Least Angle Regression (LARS) algorithm (Efron et al., 2004) is able to efficiently compute the optimal solution.

The rest of this paper is organized as follows. In Section 2 we present preliminaries and review the LSTD algorithm. Section 3 contains the main contribution of this paper: we present a regularized version of the LSTD algorithm and give an efficient algorithm for solving the l_1 regularized case. In Section 4 we present experimental results. Finally, in Section 5 we discuss relevant related work and conclude the paper in Section 6.

2. Background and Preliminaries

A Markov Decision Process (MDP) is a tuple (S, A, P, d, R, γ), where S is a set of states; A is a set of actions; $P : S \times A \times S \to [0, 1]$ is a state transition probability function where $P(s, a, s')$ denotes the probability of transitioning to state s' when taking action a from state s; d is a distribution over initial states; $R : S \to \mathbb{R}$ is a reward function; and $\gamma \in [0, 1]$ is a discount factor. For simplicity of presentation, we will assume that S and A are finite, though potentially very

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009.

large; this merely permits us to use matrix rather than operator notation, though the results here hold with minor technicalities in infinite state spaces. A policy is $\pi : S \to A$ is a mapping from states to actions.

The notion of a *value function* is of central importance in reinforcement learning; in our setting, for a given policy π, the value of a state s is defined as the expected discounted sum of rewards obtained when starting in state s and following policy π: $V^\pi(s) = E[\sum_{t=0}^{\infty} \gamma^t R(s_t)|s_0 = s, \pi]$. It is well-known that the value function must obey *Bellman's equation*

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

or expressed in vector form

$$V^\pi = R + \gamma P^\pi V^\pi \tag{1}$$

where $V, R \in \mathbb{R}^{|S|}$ are vectors containing the state values and rewards respectively, and $P^\pi \in \mathbb{R}^{|S|\times|S|}$ is a matrix encoding the transitions probabilities of the policy π, $P^\pi_{i,j} = P(s' = j|s = i, \pi(s))$. If both the rewards and transition probabilities are known, then we can solve for the value function analytically by solving the linear system $V^\pi = (I - \gamma P^\pi)^{-1} R$.

However, in the setting that we consider in this paper, the situation is significantly more challenging. First, we consider a setting where the transition probability matrix is not known, but where we only have access to a *trajectory*, a sequence of states $s_0, s_1, \ldots$ where $s_0 \sim d$ and $s_{i+1} \sim P(s_i, \pi(s_i))$. Second, as mentioned previously, we are explicitly interested in the setting where the number of states is large enough that the value function cannot be expressed explicitly, and so instead we must resort to function approximation. We focus on the case of linear function approximation, i.e.,

$$V^\pi(s) \approx w^T \phi(s)$$

where $w \in \mathbb{R}^k$ is a parameter vector and $\phi(s) \in \mathbb{R}^k$ is a feature vector corresponding to the state s. Again adopting vector notation, this can be written $V^\pi \approx \Phi w$ where $\Phi \in \mathbb{R}^{|S|\times k}$ is a matrix whose rows contains the feature vectors for every state. Unfortunately, when approximating V^π in this manner, there is usually no way to satisfy the Bellman equation (1) exactly, because the vector $R + \gamma P^\pi \Phi w$ may lie outside the span of the bases Φ.

2.1. Review of Least-Squares Temporal Difference Methods

The Least-Squares Temporal Difference (LSTD) algorithm presents a method for finding the parameters w such that the resulting value function "approximately" satisfies the Bellman equation. Although $R + \gamma P^\pi \Phi w$ may not lie in the span of the bases, we can find the closest approximation to this vector that *does* lie in the span of the bases by solving the least-squares problem,

$$\min_{u\in\mathbb{R}^k} \|\Phi u - (R + \gamma P^\pi \Phi w)\|_D^2 \tag{2}$$

where D is a non-negative diagonal matrix indicating a distribution over states.[1] The TD family of algorithms in general, including LSTD, attempt to find a fixed point of the above operation; that is, they attempt to find w such that

$$w = f(w) = \arg\min_{u\in\mathbb{R}^k} \|\Phi u - (R + \gamma P^\pi \Phi w)\|_D^2. \tag{3}$$

We now briefly review the LSTD algorithm. We first note that since P^π is unknown, and since the full Φ matrices are too large to form anyway, we cannot solve (3) exactly. Instead, given a trajectory consisting of states, actions, and next-states, $(s_i, a_i, s'_i), i = 1 \ldots m$ collected from the MDP of interest, we define the sample matrices

$$\tilde{\Phi} \equiv \begin{bmatrix} \phi(s_1)^T \\ \vdots \\ \phi(s_m)^T \end{bmatrix}, \; \tilde{\Phi}' \equiv \begin{bmatrix} \phi(s'_1)^T \\ \vdots \\ \phi(s'_m)^T \end{bmatrix}, \; \tilde{R} \equiv \begin{bmatrix} r_1 \\ \vdots \\ r_m \end{bmatrix}. \tag{4}$$

Given these samples, LSTD finds a fixed point of the approximation

$$w = \tilde{f}(w) = \arg\min_{u\in\mathbb{R}^k} \|\tilde{\Phi} u - (\tilde{R} + \gamma \tilde{\Phi}' w)\|^2. \tag{5}$$

It is straightforward to show that with probability one, as the number of samples $m \to \infty$, the fixed point of the approximation (5) equals the fixed point of the true equation (3), where the diagonal entries of D are equal to the distribution over samples (Bradtke & Barto, 1996).

In addition, since the minimization in (5) contains only a Euclidean norm, the optimal u can be computed analytically as

$$\tilde{f}(w) = u^\star = (\tilde{\Phi}^T \tilde{\Phi})^{-1} \tilde{\Phi}^T (\tilde{R} + \gamma \tilde{\Phi}' w).$$

Now we can find the fixed point $w = \tilde{f}(w)$ by simply solving a linear system

$$\begin{aligned} w &= (\tilde{\Phi}^T \tilde{\Phi})^{-1} \tilde{\Phi}^T (\tilde{R} + \gamma \tilde{\Phi}' w) \\ &= \left(\tilde{\Phi}^T (\tilde{\Phi} - \gamma \tilde{\Phi}')\right)^{-1} \tilde{\Phi}^T \tilde{R} = \tilde{A}^{-1} \tilde{b} \end{aligned}$$

[1] The squared norm $\|\cdot\|_D^2$ is defined as $\|x\|_D^2 = x^T D x$.

where we define $\tilde{A} \in \mathbb{R}^{k\times k}$ and $\tilde{b} \in \mathbb{R}^k$ as

$$\begin{aligned}\tilde{A} &\equiv \tilde{\Phi}^T(\tilde{\Phi} - \gamma\tilde{\Phi}') = \sum_{i=1}^{m} \phi(s_i)\left(\phi(s_i) - \gamma\phi(s_i')\right)^T \\ \tilde{b} &\equiv \tilde{\Phi}^T\tilde{R} = \sum_{i=1}^{m} \phi(s_i) r_i.\end{aligned} \tag{6}$$

Given a collection of samples, the LSTD algorithm forms the $\tilde{A}$ and $\tilde{b}$ matrices using above formula, then solves the $k\times k$ linear system $w = \tilde{A}^{-1}\tilde{b}$. When k is relatively small solving this linear system is very fast, so if there are a sufficiently large number of samples to estimate these matrices, the algorithm can perform very well. In particular LSTD has been demonstrated to make more efficient use of samples than the standard TD algorithm, and it requires no tuning of a learning rate or initial guess of w (Bradtke & Barto, 1996; Boyan, 2002).

Despite its advantages, LSTD also has several drawbacks. First, if the number of basis functions is very large, then LSTD will require a prohibitively large number of samples in order to obtain a good estimate of the parameters w; if there is too little data available, then it can over-fit significantly or fail entirely — for example, if $m < k$, then the matrix $\tilde{A}$ will not be full rank. Furthermore, since LSTD requires storing and inverting a $k \times k$ matrix, the method is not feasible if k is large (there exist extensions to LSTD that alleviate this problem slightly from a run-time perspective (Geramifard et al., 2006), but the methods still require a prohibitively large amount of storage).

3. Temporal Difference with Regularized Fixed Points

In this section we present a regularization framework that allows us to overcome the difficulties discussed in the previous section. While the general idea of applying regularization for feature selection and to avoid over-fitting is of course a common theme in machine learning and statistics, applying it to the LSTD algorithm is challenging due to the fact that this algorithm is based on finding a fixed-point rather than optimizing some convex objective.

We begin by augmenting the fixed point function of the LSTD algorithm (5) to include a regularization term

$$\tilde{f}(w) = \arg\min_{u\in\mathbb{R}^k} \frac{1}{2}\|\tilde{\Phi}u - (\tilde{R} + \gamma\tilde{\Phi}'w)\|^2 + \beta\nu(u) \tag{7}$$

where $\beta \in [0, \infty)$ is a regularization parameter and $\nu : \mathbb{R}^k \to \mathbb{R}_+$ is a regularization penalty function — in this work, we specifically consider l_2 and l_1 regularization corresponding to $\nu(u) = \frac{1}{2}\|u\|_2^2$ and $\nu(u) = \|u\|_1$ respectively (or possibly a combination of the two). With this modification, it is no longer immediately clear if there exist fixed points $w = \tilde{f}(w)$ for all β, and it is also unclear how we may go about finding this fixed point, if it exists.

3.1. l_2 Regularization

The case of l_2 regularization is fairly straightforward, but we include it here for completeness. When $\nu(u) = \frac{1}{2}\|u\|_2^2$, the optimal u can again be solved for in closed form, as in standard LSTD. In particular, it is straightforward to show that

$$\tilde{f}(w) = (\tilde{\Phi}^T\tilde{\Phi} + \beta I)^{-1}\tilde{\Phi}^T(\tilde{R} + \gamma\tilde{\Phi}'w)$$

so the fixed point $w = \tilde{f}(w)$ can be found by

$$w = \left(\tilde{\Phi}^T(\tilde{\Phi} - \gamma\tilde{\Phi}') + \beta I\right)^{-1}\tilde{\Phi}^T\tilde{R} = (\tilde{A} + \beta I)^{-1}\tilde{b}.$$

Clearly, such a fixed point exists with probability 1, since $A + \beta I$ is invertible unless one of the eigenvalues of A is equal to $-\beta$, which constitutes a set of measure zero. Indeed, many practical implementations of LSTD already implement some regularization of this type to avoid the possibility of singular $\tilde{A}$. However, this type of regularization does not address the concerns from the previous section: we still need to form and invert the entire $k \times k$ matrix, and as we will show in the next section, this method still performs poorly when the number of samples is small compared to the number of features.

3.2. l_1 Regularization

We now come to the primary algorithmic contribution of the paper, a method for finding the fixed point of (7) when using l_1 regularization, $\nu(u) = \|u\|_1$. Since l_1 regularization is known to produce sparse solutions, it can both allow for more efficient implementation and be effective in the context of feature selection — many experimental and theoretical results confirm this assertion in the context of supervised learning (Tibshirani, 1996; Ng, 2004). To give a brief overview of the main ideas in this section, it will turn out that finding the l_1 regularized fixed point cannot be expressed as a convex optimization problem. Nonetheless, it is possible to adapt an algorithm known as Least Angle Regression (LARS) (Efron et al., 2004) to this task.

Space constrains preclude a full discussion, but briefly, the the LARS algorithm is a method for solving l_1 regularized problem least-squares problems

$$\min_w \|Aw - b\|^2 + \beta\|w\|_1.$$

Although the l_1 objective here is non-differentiable, ruling out an analytical solution like the l_2 regular-

ized case, it turns out that the optimal solution w can be built incrementally, updating one element of w at a time, until we reach the exact solution of the optimization problem. This is the basic idea behind the LARS algorithm, and in the remainder of this section, we will show how this same intuition can be applied to find l_1 regularized fixed points of the TD equation.

To begin, we transform the l_1 optimization problem (7) into a set of optimality conditions, following e.g. Kim et al. (2007) — these optimality conditions can be derived from sub-differentials, but the precise derivation is unimportant

$$\begin{aligned} -\beta \leq (\tilde{\Phi}^T((\tilde{R} + \gamma\tilde{\Phi}'w) - \tilde{\Phi}u))_i \leq \beta \qquad \forall i \\ (\tilde{\Phi}^T((\tilde{R} + \gamma\tilde{\Phi}'w) - \tilde{\Phi}u))_i = \beta \Rightarrow u_i \geq 0 \\ (\tilde{\Phi}^T((\tilde{R} + \gamma\tilde{\Phi}'w) - \tilde{\Phi}u))_i = -\beta \Rightarrow u_i \leq 0 \\ -\beta < (\tilde{\Phi}^T((\tilde{R} + \gamma\tilde{\Phi}'w) - \Phi u))_i < \beta \Rightarrow u_i = 0. \end{aligned} \tag{8}$$

Since the optimization problem (7) is convex, these conditions are both necessary and sufficient for the global optimality of a solution u — i.e., if we can find some u satisfying these conditions, then it is a solution to the optimization problem. Therefore, in order for a point w to be a fixed point of the equation (7) with l_1 regularization, it is necessary and sufficient that the above equations hold for $u = w$ — i.e., the following optimality conditions must hold

$$\begin{aligned} -\beta \leq (\tilde{\Phi}^T\tilde{R} - \tilde{\Phi}^T(\tilde{\Phi} - \gamma\tilde{\Phi}')w)_i \leq \beta \qquad \forall i \\ (\tilde{\Phi}^T\tilde{R} - \tilde{\Phi}^T(\tilde{\Phi} - \gamma\tilde{\Phi}')w)_i = \beta \Rightarrow w_i \geq 0 \\ (\tilde{\Phi}^T\tilde{R} - \tilde{\Phi}^T(\tilde{\Phi} - \gamma\tilde{\Phi}')w)_i = -\beta \Rightarrow w_i \leq 0 \\ -\beta < (\tilde{\Phi}^T\tilde{R} - \tilde{\Phi}^T(\tilde{\Phi} - \gamma\tilde{\Phi}')w)_i < \beta \Rightarrow w_i = 0. \end{aligned} \tag{9}$$

It is also important to understand that the substitutions performed here are *not* the same as solving the optimization problem (7) with the additional constraint that $u = w$; rather, we first find the optimality conditions of the optimization problem and *then* substitute $u = w$. Indeed, because $\tilde{\Phi}^T(\tilde{\Phi} - \gamma\tilde{\Phi}')$ is not symmetric, the optimality conditions (9) do not correspond to *any* optimization problem, let alone a convex one. Nonetheless, as we will show, we are still able to find solutions to this optimization problem. The resulting algorithm is very efficient, since we form only those rows and columns of the $\tilde{A}$ matrix corresponding to non-zero coefficients in w, which is typically sparse.

We call our algorithm LARS-TD to highlight the connection to LARS and give pseudo-code for the algorithm in Figure 1.[2] The algorithm maintains an active set $\mathcal{I} = \{i_1, \ldots, i_{|\mathcal{I}|}\}$, corresponding to the non-zero coefficients w. At each step, the algorithm obtains a solution to the optimality conditions (9) for some $\bar{\beta} \geq \beta$, and continually reduces this bound until it is equal to β.

Algorithm LARS-TD($\{s_i, r_i, s_i'\}, \phi, \beta, \gamma$)

Parameters:

$\{s_i, r_i, s_i'\}, i = 1, \ldots, m$: state transition and reward samples

$\phi : S \rightarrow \mathbb{R}^k$ value function basis

$\beta \in \mathbb{R}_+$: regularization parameter

$\gamma \in [0, 1]$: discount factor

Initialization:

1. Set $w \leftarrow 0$ and initialize the correlation vector $c \leftarrow \sum_{i=1}^m \phi(s_i) r_i$.
2. Let $\{\bar{\beta}, i\} \leftarrow \max_j\{|c_j|\}$ and initialize the active set $\mathcal{I} \leftarrow \{i\}$.

While $(\bar{\beta} > \beta)$**:**

1. Find update direction $\Delta w_{\mathcal{I}}$:
   $\Delta w_{\mathcal{I}} \leftarrow \tilde{A}_{\mathcal{I},\mathcal{I}}^{-1}\text{sign}(c_{\mathcal{I}})$
   $\tilde{A}_{\mathcal{I},\mathcal{I}} \equiv \sum_{i=1}^m \phi_{\mathcal{I}}(s_i)\left(\phi_{\mathcal{I}}(s_i) - \gamma\phi_{\mathcal{I}}(s_i')\right)^T$
2. Find step size to add element to the active set:
   $\{\alpha_1, i_1\} \leftarrow \min^+_{j \notin \mathcal{I}} \left\{ \frac{c_j - \bar{\beta}}{d_j - 1}, \frac{c_j + \bar{\beta}}{d_j + 1} \right\}$
   $d \equiv \sum_{i=1}^m \phi(s_i)\left(\phi_{\mathcal{I}}(s_i) - \gamma\phi_{\mathcal{I}}(s_i')\right)^T \Delta w_{\mathcal{I}}$
3. Find step size to reach zero coefficient:
   $\{\alpha_2, i_2\} \leftarrow \min^+_{j \in \mathcal{I}} \left\{ -\frac{w_j}{\Delta w_j} \right\}$
4. Update weights, $\bar{\beta}$, and correlation vector:
   $w_{\mathcal{I}} \leftarrow w_{\mathcal{I}} + \alpha\Delta w_{\mathcal{I}}$, $\bar{\beta} \leftarrow \bar{\beta} - \alpha$, $c \leftarrow c - \alpha d$ where $\alpha = \min\{\alpha_1, \alpha_2, \bar{\beta} - \beta\}$.
5. Add i_1 or remove i_2 from active set:
   If $(\alpha_1 < \alpha_2)$, $\mathcal{I} \leftarrow \mathcal{I} \bigcup \{i_1\}$
   else, $\mathcal{I} \leftarrow \mathcal{I} - \{i_2\}$.

Return w.

Figure 1. The LARS-TD algorithm for l_1 regularized LSTD. See text for description and notation.

We describe the algorithm inductively. Suppose that at some point during execution, we have the index set $\mathcal{I}$ and corresponding coefficients $w_{\mathcal{I}}$ which satisfy the optimality conditions (9) for some $\bar{\beta} > \beta$ — note that the vector $w = 0$ is guaranteed to satisfy this condition for some $\bar{\beta}$. We define the "correlation coefficients" (or just "correlations") $c \in \mathbb{R}^k$ as

$$c \equiv \tilde{\Phi}^T\tilde{R} - \tilde{\Phi}^T(\tilde{\Phi} - \gamma\tilde{\Phi}')w = \tilde{\Phi}^T\tilde{R} - \tilde{\Phi}^T(\tilde{\Phi}_{\mathcal{I}} - \gamma\tilde{\Phi}'_{\mathcal{I}})w$$

where for a vector or matrix $x_{\mathcal{I}}$ denotes the rows of x corresponding to the indices in $\mathcal{I}$.

By the optimality conditions, we know that $c_{\mathcal{I}} = \pm\bar{\beta}$ and that $|c_i| < \bar{\beta}$ for all $i \notin \mathcal{I}$. Therefore, when updating w we must ensure that all the $c_{\mathcal{I}}$ terms are adjusted

[2] Here and in the text $\min^+$ denotes the minimum taken only over non-negative elements. In addition $\{x, i\} \leftarrow \min$ indicates that x should take the value of the minimum element, while i takes the value of the corresponding index.

equally, or the optimality conditions will be violated. This leads to the update direction

$$\Delta w = \left(\tilde{\Phi}_{\mathcal{I}}^T(\tilde{\Phi}_{\mathcal{I}} - \gamma\tilde{\Phi}'_{\mathcal{I}})\right)^{-1} \text{sign}(c_{\mathcal{I}})$$

where $\text{sign}(c_{\mathcal{I}})$ denotes the vector of $\{-1, +1\}$ entries corresponding to the signs of $c_{\mathcal{I}}$. Given this update direction for w, we take as large a step in this direction as possible until some c_i for $i \notin \mathcal{I}$ also reaches the bound. This step size can be found analytically as

$$\alpha_1 = \min_{i \notin \mathcal{I}}{}^+ \left\{ \frac{c_i - \bar{\beta}}{d_i - 1}, \frac{c_i + \bar{\beta}}{d_i + 1} \right\},$$

where

$$d \equiv \tilde{\Phi}^T(\tilde{\Phi}_{\mathcal{I}} - \gamma\tilde{\Phi}'_{\mathcal{I}})\Delta w$$

(here d indicates how much a step in the direction Δw will affect the correlations c). At this point c_i is at the bound, so we add i to the active set $\mathcal{I}$.

Lastly, the optimality conditions requires that for all i, the signs of the coefficients agree with the signs of the correlations. To prevent this condition from being violated we also look for any points along the update direction where the sign of any coefficient changes; the smallest step-size for which this occurs is given by

$$\alpha_2 = \min_{i \in \mathcal{I}}{}^+ \left\{ -\frac{w_i}{\Delta w_i} \right\}$$

(if no such positive elements exist, then we take $\alpha_2 = \infty$). If $\alpha_1 < \alpha_2$, then no coefficients would change sign during our normal update, so we proceed as before. However, if $\alpha_2 < \alpha_1$, then we update w with a step size of α_2, and remove the corresponding zero coefficient from the active set. This completes the description of the LARS-TD algorithm.

Computational complexity and extensions to LSTD(λ) and LSTDQ. One iteration of the LARS-TD algorithm presented above has time complexity $O(mkp^3)$ — or $O(mkp^2)$ if we use an LU factorization update/downdate to invert the $\tilde{A}_{\mathcal{I},\mathcal{I}}$ matrix — and space complexity $O(k + p^2)$ where p is the number of non-zero coefficients of w, m is the number of samples, and k is the number of basis functions. In practice, the algorithm typically requires a number of iterations that is about equal to some constant factor times the final number of active basis functions, so the total time complexity of an efficient implementation will be approximately $O(mkp^3)$. The crucial property here is that the algorithm is *linear* in the number of basis function and samples, which is especially important in the typical case where $p \ll m, k$.

For the sake of clarity, the algorithm we have presented so far is a regularized generalization of the LSTD(0) algorithm. However, our algorithm can be extended to LSTD(λ) (Boyan, 2002) by the use of eligibility traces, or to LSTDQ (Lagoudakis & Parr, 2003), which learns state-action value functions $Q(s, a)$. Space constraints preclude a full discussion, but the extensions are straightforward.

3.3. Correctness of LARS-TD, P-matrices, and the continuity of l_1 fixed points

The following theorem shows that LARS-TD finds an l_1 regularized fixed point under suitable conditions. Due to space constraints, the full proof is presented in an appendix, available in the full version of the paper (Kolter & Ng, 2009). However, the algorithm is *not* guaranteed to find a fixed point for the specified value of β in every case, though we have never found this to be a problem in practice. A sufficient condition for LARS-TD to find a solution for any value of β is for $\tilde{A} = \tilde{\Phi}^T(\tilde{\Phi} - \gamma\tilde{\Phi}')$ to be a *P-matrix*[3] as formalized by the following theorem.

Theorem 3.1 *If $\tilde{A} = \tilde{\Phi}^T(\tilde{\Phi} - \gamma\tilde{\Phi}')$ is a P-matrix, then for any $\beta \geq 0$, the LARS-TD algorithm finds a solution to the l_1 regularized fixed-point optimality conditions* (9).

Proof (Outline) The proof follows by a similar inductive argument as we used to describe the algorithm, but requires two conditions that we show to hold when $\tilde{A}$ is a P-matrix. First, we must guarantee that when an index i is added to the active set, the sign of its coefficient is equal to the sign of its correlation. Second, if an index i is removed from the active set, its correlation will still be at bound initially and so its correlation must be decreased at a faster rate than the correlations in the active set. □

A simple two-state Markov chain, shown in Figure 2(a), demonstrates why LARS-TD may not find the solution when $\tilde{A}$ is not a P-matrix. Suppose that we run the LARS-TD algorithm using a single basis function $\Phi = [10.0 \;\; 2.0]^T$, $\gamma = 0.95$, and uniform sampling $D = \text{diag}(0.5, 0.5)$. We can then compute A and b

$$A = \Phi^T D(\Phi - \gamma P^\pi \Phi) = -5.0, \quad b = \Phi^T DR = 5.0,$$

[3] A matrix $A \in \mathbb{R}^{n \times n}$ is a P-matrix if all its principle minors (the determinants of sub-matrices formed by taking some subset of the rows and columns) are positive (Horn & Johnson, 1991). The class of P-matrices is a strict superset of the class of positive definite (including non-symmetric positive definite) matrices, and we use the former in order to include various cases not covered by the positive definite matrices alone. For example, in the case that the bases are simply the identity function or a grid, $\Phi^T(\Phi - \gamma P^\pi \Phi) = (I - \gamma P)$, and while this matrix is often not positive definite, it is always a P-matrix.

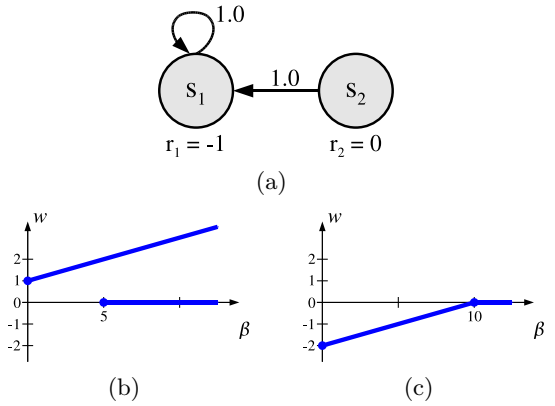

Figure 2. (a) The MDP used to illustrate the possibility that LARS-TD does not find a fixed point. (b) Plot indicating all the fixed points of the regularized TD algorithm given a certain basis and sampling distribution. (c) The fixed points for the same algorithm when the sampling is on-policy.

so A matrix is not a P-matrix. Figure 2(b) shows all the fixed points for the l_1 regularized TD equation (7). Note that there are multiple fixed point for a given value of β, and that there does not exist a continuous path from the null solution $w = 0$ to the LSTD solution $w = 1$; this renders LARS-TD unable to find the fixed points for all β.

There are several ways to ensure that $\tilde{A}$ is a P-matrix and, as mentioned, we have never found this to be a problem in practice. First, as shown in (Tsitsiklis & Roy, 1997), when the sampling distribution is *on-policy* in that D equals the stationary distribution of the Markov chain, A (and therefore $\tilde{A}$, given enough samples) is positive definite and therefore a P-matrix; in our chain example, this situation is demonstrated in Figure 2(c). However, even when sampling off-policy we can also ensure that $\tilde{A}$ is a P-matrix by additionally adding some amount of l_2 regularization; this is known as elastic net regularization (Zou & Hastie, 2005). Finally, we can check if the optimality conditions are violated at each step along the regularization path and simply terminate if continuous path exists from the current point to the LSTD solution. This general technique of stopping a continuation method if it reaches a discontinuity has been proposed in other settings as well (Corduneanu & Jaakkola, 2003).

4. Experiments

4.1. Chain domain with irrelevant features

We first consider a simple discrete 20-state "chain" MDP, proposed in (Lagoudakis & Parr, 2003), to demonstrate that l_1 regularized LSTD can cope with many irrelevant features. The MDP we use consists of 20 states, two actions, left and right, and a reward of one at each of the ends of the chain. To represent

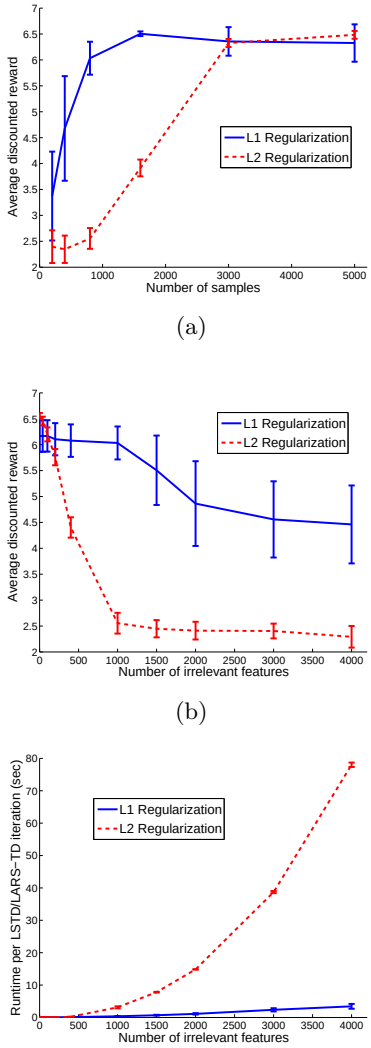

Figure 3. (a) Average reward versus number of samples for 1000 irrelevant features on the chain domain. (b) Average reward versus number of irrelevant features for 800 samples. (c) Run time versus number of irrelevant features for 800 samples.

the value function, we used six "relevant" features — five RBF basis functions spaced evenly across the domain and a constant term — as well as some number of irrelevant noise features, just containing Gaussian random noise for each state. To find the optimal policy, we used the LSPI algorithm with the LARS-TD algorithm, modified to learn the Q function. Regularization parameters for both the l_1 and l_2 cases were found by testing a small number of regularization parameters on randomly generated examples, though the algorithms performed similarly for a wide range of parameters. Using no regularization at all — i.e., standard LSPI — performed worse in all cases. All results above were averaged over 20 runs, and we report 95% confidence intervals.

Table 1. Success probabilities and run times for LARS-TD and l_2 LSTD on the mountain car.

Algorithm	**LARS-TD**	**l_2 LSTD**
Success %	100% (20/20)	0% (0/20)
Iteration Time (sec)	1.20 ± 0.27	3.42 ± 0.04

As shown in Figures 3(a) and 3(b) both l_2 and l_1 regularized LSTD perform well when there are no irrelevant features, but l_1 performs significantly better using significantly less data in the presence of many irrelevant features. Finally, Figure 3(c) shows the run-time of one iteration of LSTD and LARS-TD. For 2000 irrelevant features, the runtime of LARS-TD is more than an order of magnitude less than that of l_2 regularized LSTD, in addition to achieving better performance; furthermore, the time complexity of LARS-TD is growing roughly linearly in the total number of features, confirming the computational complexity bound presented earlier.

4.2. Mountain car

We next consider the classic "mountain car" domain (Sutton & Barto, 1998), consisting of a continuous two-dimensional state space. In continuous domains, practitioners frequently handcraft basis functions to represent the value function; a common choice is to use radial basis functions (RBFs), which are evenly spaced grids of Gaussian functions over the state space. However, picking the right grid, spacing, etc, of the RBFs is crucial to obtain good performance, and often takes a significant amount of hand-tuning. Here we use the mountain car domain to show that our proposed l_1 regularization algorithm can alleviate this problem significantly: we simply use many different sets of RBFs in the problem, and let the LARS-TD algorithm pick the most relevant. In particular, we used two-dimensional grids of 2, 4, 8, 16, and 32 RBFs, plus a constant offset, for a total of 1365 basis functions. We then collected up to 500 samples by executing 50 episodes starting from a random state and executing a random policy for up to 10 time steps. Using only this data, we used policy iteration and off-policy LSTD/LARS-TD to find a policy.

Table 1 summarizes the results: despite the fact that we have a relatively small amount of training data and many basis functions, LARS-TD is able to find a policy that successfully brings the car up the hill 100% of the time (out of 20 trials). As these policies are learned from very little data, they are not necessarily optimal: they reach the goal in an average of 142.25 ± 9.74 steps, when starting from the initial state. However, in contrast, LSTD with l_2 or no regularization at all is never able to find a successful policy.

5. Related Work

In addition to the work on least-squares temporal difference methods as well as l_1 regularization methods that we have mentioned already in this paper, there has been some recent work on regularization and feature selection in Reinforcement Learning. For instance, (Farahmand et al., 2009) consider a regularized version of TD-based policy iteration algorithms, but only specifically consider l_2 regularization — their specific scheme for l_2 regularization varies slightly from ours, but the general idea is quite similar to the l_2 regularization we present. However, the paper focuses mainly on showing how such regularization can guarantee theoretical convergence properties for policy iteration, which is a mainly orthogonal issue to the l_1 regularization we consider here.

Another class of methods that bear some similarity to our own are recent methods for feature generation based on the Bellman error (Menache et al., 2005; Keller et al., 2006; Parr et al., 2007). In particular (Parr et al., 2007) analyze approaches that continually add new basis functions to the current set, based upon their correlation with the Bellman residual. The comparison between this work and our own is roughly analogous to the comparison between the classical "forward-selection" feature-selection method and l_1 based feature selection methods; these two general methods are compared in the supervised least-squares setting by Efron et al. (2004), and the l_1 regularized approach is typically understood to have better statistical estimation properties.

The algorithm presented in (Loth et al., 2007) bears a great deal similarity to our own, as they also work with a form of l_1 regularization. However, the details of the algorithm are quite different: the authors do not consider fixed points of a Bellman backup, but instead just optimize the distance to the standard LSTD solution plus an additional regularization term. The solution then loses all interpretation as a fixed point, which has proved very important for the TD solution, and also loses much of the computational benefit of LARS-TD, since, using the notation from Section 3, they need to compute entire columns of the $\tilde{A}^T\tilde{A}$ matrix. Therefore, it is unclear whether this approach can be implemented efficiently (i.e., in time and memory linear in the number of features). Furthermore, this previous work focuses mainly on kernel features, so that the approach is more along the lines of those discussed in the next paragraph, where the primary goal is selecting a sparse subset of the *samples*.

There has also been work on feature selection and sparsification in kernelized RL, but this work is only tangentially related to our own. For instance, in (Xu

et al., 2007), the authors present a kernelized version of the LSPI algorithm, and also include a technique to sparsify the algorithm — a similar approach is also presented in (Jung & Polani, 2006). However, the motivation and nature of their algorithm is quite different from ours and not fully comparable: as they are working in the kernel domain, the primary concern is obtaining sparsity in the *samples* used in the kernelized solution (similar to support vector machines), and they use a simple greedy heuristic, whereas our algorithm obtains sparsity in the *features* via regularization.

6. Conclusion

In this paper we proposed a regularization framework for least-squares temporal difference learning. In particular, we proposed a method for finding the temporal difference fixed point augmented with an l_1 regularization term. This type of regularization is an effective method for feature selection in reinforcement learning, and we demonstrated this experimentally.

Acknowledgments

This work was supported by the DARPA Learning Locomotion program under contract number FA8650-05-C-7261. We thank the anonymous reviews for helpful comments. Zico Kolter is partially supported by an NSF Graduate Research Fellowship.

References

Boyan, J. (2002). Technical update: Least-squares temporal difference learning. *Machine Learning*, *49*, 233–246.

Bradtke, S., & Barto, A. (1996). Linear least-squares algorithms for temporal difference learning. *Machine Learning*, *22*, 33–57.

Corduneanu, A., & Jaakkola, T. (2003). On information regularization. *Proceedings of the Conference on Uncertainty in Artificial Intelligence* (pp. 151–158).

Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression. *Annals of Statistics*, *32*, 407–499.

Farahmand, A. M., Ghavamzadeh, M., Szepesvari, C., & Mannor, S. (2009). Regularized policy iteration. *Neural Information Processing Systems* (pp. 441–448).

Geramifard, A., Bowling, M., & Sutton, R. (2006). Incremental least-squares temporal difference learning. *Proceedings of the American Association for Artitical Intelligence* (pp. 356–361).

Horn, R. A., & Johnson, C. R. (1991). *Topics in matrix analysis*. Cambridge University Press.

Jung, T., & Polani, D. (2006). Least squares svm for least squares td learning. *Proceedings of the European Conference on Artificial Intelligence* (pp. 499–503).

Keller, P. W., Mannor, S., & Precup, D. (2006). Automatic basis function construction for approximate dynamic programming and reinforcement learning. *Proceedings of the International Conference on Machine Learning* (pp. 449–456).

Kim, S., Koh, K., Lustig, M., Boyd, S., & Gorinevsky, D. (2007). An interior-point method for large-scale l1-regularized least squares. *IEEE Journal on Selected Topics in Signal Processing*, *1*, 606–617.

Kolter, J. Z., & Ng, A. Y. (2009). Regularization and feature selection in least-squares temporal difference learning (full version). *Available at http://ai.stanford.edu/~kolter*.

Lagoudakis, M., & Parr, R. (2003). Least-squares policy iteration. *Journal of Machine Learning Research*, *4*, 1107–1149.

Loth, M., Davy, M., & Preux, P. (2007). Sparse temporal difference learning using lasso. *Proceedings of the IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning* (pp. 352–359).

Menache, I., Mannor, S., & Shimkin, N. (2005). Basis function adaptation in temporal difference reinforcement learning. *Annals of Operations Research*, *134*, 215–238.

Ng, A. Y. (2004). Feature selection, l1 vs. l2 regularization, and rotational invariance. *Proceedings of the International Conference on Machine Learning*.

Parr, R., Painter-Wakefield, C., Li, L., & Littman, M. (2007). Analyzing feature generation for value-function approximation. *Proceedings of the International Conference on Machine Learing* (pp. 737–744).

Sutton, R. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, *3*, 9–44.

Sutton, R., & Barto, A. (1998). *Reinforcement learning: An introduction*. MIT Press.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society B*, *58*, 267–288.

Tsitsiklis, J., & Roy, B. V. (1997). An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, *42*, 674–690.

Xu, X., Hu, D., & Lu, X. (2007). Kernel-based least squares policy iteration for reinforcement learning. *IEEE Transactions on Neural Networks*, *18*, 973–992.

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society B*, *67*, 301–320.