Regularized Least Squares Temporal Difference learning with nested ℓ_2 and ℓ_1 penalization

Matthew W. Hoffman[1], Alessandro Lazaric[2], Mohammad Ghavamzadeh[2], and Rémi Munos[2]

[1] University of British Columbia, Computer Science, Vancouver, Canada
[2] INRIA Lille - Nord Europe, Team SequeL, France

Abstract. The construction of a suitable set of features to approximate value functions is a central problem in reinforcement learning (RL). A popular approach to this problem is to use high-dimensional feature spaces together with least-squares temporal difference learning (LSTD). Although this combination allows for very accurate approximations, it often exhibits poor prediction performance because of overfitting when the number of samples is small compared to the number of features in the approximation space. In the linear regression setting, regularization is commonly used to overcome this problem. In this paper, we review some regularized approaches to policy evaluation and we introduce a novel scheme (L_{21}) which uses ℓ_2 regularization in the projection operator and an ℓ_1 penalty in the fixed-point step. We show that such formulation reduces to a standard Lasso problem. As a result, any off-the-shelf solver can be used to compute its solution and standardization techniques can be applied to the data. We report experimental results showing that L_{21} is effective in avoiding overfitting and that it compares favorably to existing ℓ_1 regularized methods.

1 Introduction

In the setting of reinforcement learning (RL), least-squares temporal difference learning (LSTD) [2] is a very popular mechanism for approximating the value function V^π of a given policy π. More precisely, this approximation is accomplished using a linear function space $\mathcal{F}$ spanned by a set of k features $\{\phi_i\}_{i=1}^k$. Here, the choice of feature set greatly determines the accuracy of the value function estimate. In practice, however, there may be no good, *a priori* method of selecting these features. One solution to this problem is to use high-dimensional feature spaces in the hopes that a good set of features lies somewhere in this basis. This introduces other problems, though, as the number of features can outnumber the number of samples $n \leq k$, leading to overfitting and poor prediction. In the linear regression setting, regularization is commonly used to overcome this problem. The two most common regularization approaches involve using ℓ_1 or ℓ_2 penalized least-squares, known as *Lasso* or *ridge regression* respectively (see e.g., [7]). The approach of Lasso is of particular interest due to its *feature selection* property, wherein the geometric form of the ℓ_1 penalty tends to encourage sparse solutions. This property is especially beneficial in high-dimensional problems, as they allow solutions to be expressed as a linear combination of a small

number of features. The application of the Lasso to the problem of value function approximation in high dimensions would thus seem to be a perfect match. However, the RL setting differs greatly from regression in that the objective is not to recover a target function given its noisy observations, but is instead to approximate the fixed-point of the Bellman operator given sample trajectories. The addition of this fixed-point creates difficulties when attempting to extend Lasso results to the RL setting. Despite these difficulties, one particular form of ℓ_1 penalization has been previously studied, both empirically [10, 9] and theoretically [8], with interesting results.

In this paper, we consider the two-level nested formulation of LSTD used in [1, 5], where the first level optimization is defined by the projection of a Bellman image onto the linear space $\mathcal{F}$, and the second level accounts for the fixed point part of the algorithm. This formulation allows us to define a wide range of algorithms depending on the specific implementation of the projection operator and the fixed point step. In particular, we will discuss a number of regularization methods for LSTD, two based on ℓ_2 penalties (one of which is introduced in [5]), one based on an ℓ_1 penalized projection [10], and finally we will introduce a novel approach which solves the problem via two nested optimization problems including an ℓ_2 and an ℓ_1 penalty (L_{21}). Unlike previous methods using ℓ_1 regularization in LSTD, this new approach introduces the ℓ_1 penalty in the fixed point step and this allows us to cast the problem as a standard Lasso problem. As a result, we do not need to develop any specific method to solve the corresponding optimization problem (as in [10], where a specific implementation of LARS has been defined) and we can apply general-purpose solvers to compute its solution. This additional flexibility also allows us to perform an explicit standardization step on the data similar to what is done in regression. We show in the experiments that all the methods using an ℓ_1 regularization successfully take advantage of the sparsity of V^π and avoid overfitting. Furthermore, we show that, similar to regression, standardization may improve the prediction accuracy, thus allowing L_{21} to achieve a better performance than LARSTD [10].

2 Preliminaries

We consider the standard RL framework [13] wherein a learning agent interacts with a stochastic environment by following a policy π. This interaction is modeled as a Markov Decision Process (MDP) given as a tuple $(\mathcal{X}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{X}$ is a set of states; $\mathcal{A}$ a set of actions; the transition kernel P is such that for all $x \in \mathcal{X}$ and $a \in \mathcal{A}$, $P(\cdot|x, a)$ is a distribution over $\mathcal{X}$; $r : \mathcal{X} \to \mathbb{R}$ is a reward function and $\gamma \in [0, 1]$ a discount factor. Given a deterministic policy $\pi : \mathcal{X} \to \mathcal{A}$, we denote by P^π the transition operator with kernel $P^\pi(\cdot|x) = P(\cdot|x, \pi(x))$.

The value function we are interested in learning maps x to its long-term expected value $V^\pi(x) = \mathbb{E}\big[\sum_{t=0}^{\infty} \gamma^t r(x_t) \big| x_0 = x, \pi\big]$. We can also define this quantity as the unique fixed-point of the Bellman operator $V^\pi = T^\pi V^\pi$, where the operator T^π is defined as

$$(T^\pi V)(x) = r(x) + \gamma \int_{\mathcal{X}} P(dx'|x, \pi(x)) V(x')$$

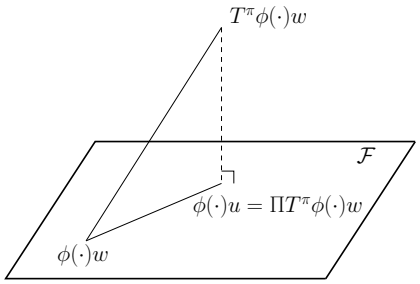

Fig. 1. A graphical illustration of the LSTD problem. Here we see the Bellman operator which takes us out of the space $\mathcal{F}$ and the orthogonal projection back onto this space.

or more concisely as $T^\pi V = r + \gamma P^\pi V$. When both r and P^π are known, this quantity can be solved for analytically, however we will consider the situation that these quantities are either unknown, too large to evaluate, or both.

Rather than directly computing the value function, we will instead seek to approximate V^π with a space $\mathcal{F}$ consisting of linear combinations of k features $\phi : \mathcal{X} \to \mathbb{R}^k$ and weights $w \in \mathbb{R}^k$, i.e. $V^\pi(x) \approx \phi(x)^T w$. The result of the Bellman operator will not necessarily lie in the span of the basis ϕ. Instead, we will adopt the approach of LSTD and approximate the resulting vector with the closest vector that *does* lie in $\mathcal{F}$. We do so by introducing a projection operator Π such that $\Pi V(x) = \phi(x)^T u^*$ where the corresponding weight is the solution to the least-squares problem: $u^* = \arg\min_{u \in \mathbb{R}^k} \|\phi^T u - V\|_\nu^2$, and where $\|f\|_\nu^2 = \int_{\mathcal{X}} f(x)^2 \nu(dx)$ is the $\ell_2(\nu)$-norm of f w.r.t. the distribution ν. By combining the Bellman and projection operators, we can write the LSTD fixed-point as $\widehat{V}^\pi = \Pi T^\pi \widehat{V}^\pi$ which, for $\widehat{V}^\pi = \phi^T w$, can be written as

$$w = u^* = \arg\min_{u \in \mathbb{R}^k} \|\phi^T u - (r + \gamma P^\pi \phi^T w)\|_\nu^2. \tag{1}$$

Alternatively, we can write the value function as the solution $\widehat{V}^\pi = \phi^T w^*$ to the following nested optimization problem [1]:

$$\begin{aligned} u^* &= \arg\min_{u \in \mathbb{R}^k} \|\phi^T u - (r + \gamma P^\pi \phi^T w)\|_\nu^2 \\ w^* &= \arg\min_{w \in \mathbb{R}^k} \|\phi^T w - \phi^T u^*\|_\nu^2. \end{aligned} \tag{2}$$

We will refer to the first problem as the *projection* step and the second as the *fixed-point* step. We should also note that the fixed-point step is not the same as the projection of $\phi^T u^*$ since u^* itself depends on w. Further, this interpretation gives us a better picture of what is being optimized by LSTD, namely that we are finding the value function $\widehat{V}^\pi$ which minimizes the distance between itself and its projected Bellman image $\Pi T^\pi \widehat{V}^\pi$ (see Figure 1).

Next, we assume an n-length trajectory consisting of transitions (x_i, a_i, r_i, x_i') sampled from the MDP of interest, and we define the sample matrices

$$\Phi = \begin{bmatrix} \phi(x_1)^T \\ \vdots \\ \phi(x_n)^T \end{bmatrix}, \; \Phi' = \begin{bmatrix} \phi(x_1')^T \\ \vdots \\ \phi(x_n')^T \end{bmatrix}, \; R = \begin{bmatrix} r(x_1) \\ \vdots \\ r(x_n) \end{bmatrix}.$$

We can then write an empirical version of (1) and solve for the fixed point by setting $w = u^*$,

$$u^* = \arg\min_{u \in \mathbb{R}^k} \|\Phi u - (R + \gamma \Phi' w)\|_2^2 = (\Phi^T \Phi)^{-1} \Phi^T (R + \gamma \Phi' w), \tag{3}$$

$$w = (\Phi^T (\Phi - \gamma \Phi'))^{-1} \Phi^T R = A^{-1} b, \tag{4}$$

where we have defined $A = \Phi^T(\Phi - \gamma\Phi')$ and $b = \Phi^T R$. While this solution provides an unbiased estimate of the value function, it can perform quite poorly when the number of samples is small in relation to the number of features. This type of scenario often results in *overfitting*, i.e. where we have more free parameters than observations, resulting in an overly complex model that is able to fit the noise of the system rather than the underlying system itself. In the next section, we examine various regularization methods designed to avoid overfitting.

3 Regularized LSTD

In this section we describe four different regularization methods which apply different penalty terms to the *projection* or to the *fixed-point* step introduced in (2). In the first two such schemes we do not penalize the fixed-point step, and we thus leave this step implicit. The final two schemes, however, rely on penalizing both sub-problems. Ultimately we describe a method which uses a mixture of ℓ_2 and ℓ_1 penalties, but that can be expressed as a standard Lasso problem.

3.1 ℓ_2 penalization (L_2)

The simplest form of regularization we can utilize involves adding an ℓ_2 penalty to the projection operator presented in (3), i.e.

$$\begin{aligned} u^* &= \arg\min_{u \in \mathbb{R}^k} \|\Phi u - (R + \gamma \Phi' w)\|_2^2 + \beta \|u\|_2^2 \\ &= (\Phi^T \Phi + \beta I)^{-1} \Phi^T (R + \gamma \Phi' w). \end{aligned} \tag{5}$$

Similar to (4), we solve for the fixed point $w = u^*$ and obtain:

$$w = (\Phi^T(\Phi - \gamma\Phi') + \beta I)^{-1} \Phi^T R = (A + \beta I)^{-1} b. \tag{6}$$

We also see here the standard LSTD components A and b, the only difference with the original formulation being the addition of β along the diagonal of A.

3.2 ℓ_1 penalization (L_1)

We can also consider adopting an ℓ_1 penalty in the projection step, i.e.

$$u^* = \arg\min_{u \in \mathbb{R}^k} \|\Phi u - (R + \gamma \Phi' w)\|_2^2 + \beta \|u\|_1. \tag{7}$$

The difficulty with this approach lies in the fact that there is now no closed-form solution to the optimization problem, a fact which causes difficulties when attempting to solve for the fixed-point $w = u^*$. Even though the projection step is just one of ℓ_1 penalized least-squares, the use of the fixed-point results in the problem not being equivalent to the Lasso. In fact, a more specialized algorithm is required to solve this problem. For a full description of this approach, and a related algorithm to solve this problem (LARSTD), we refer the reader to [10].

3.3 ℓ_2 and ℓ_2 penalization (L_{22})

Another approach we can take involves using the nested-optimization formulation of LSTD and applying regularization to both the projection and fixed-point steps. Such regularization was utilized in [5]. We can write this problem as

$$\begin{aligned} u^* &= \arg\min_{u \in \mathbb{R}^k} \|\Phi u - (R + \gamma \Phi' w)\|_2^2 + \beta \|u\|_2^2 \\ w^* &= \arg\min_{w \in \mathbb{R}^k} \|\Phi w - \Phi u^*\|_2^2 + \beta' \|w\|_2^2. \end{aligned} \tag{8}$$

Just as we did in (5) we can find a closed-form solution to u^* and can then simplify the residual term of the fixed-point subproblem as

$$\Phi w - \Phi u^* = \Phi w - \underbrace{\Phi(\Phi^T \Phi + \beta I)^{-1} \Phi^T}_{\Sigma} (R + \gamma \Phi' w). \tag{9}$$

Here the matrix Σ represents the empirical ℓ_2 penalized projection operator, or *hat matrix*, which projects n-vectors onto the space spanned by the features Φ. We can then solve for w^* in closed form as

$$w^* = \arg\min_{w \in \mathbb{R}^k} \| \overbrace{(\Phi - \gamma \Sigma \Phi')}^{X} w - \overbrace{\Sigma R}^{y} \|_2^2 + \beta' \|w\|_2^2 = (X^T X + \beta' I)^{-1} X^T y. \tag{10}$$

We can also, however, formulate this problem in terms of the standard LSTD matrices (as defined in Section 2) by noting that for $C = \Phi(\Phi^T \Phi + \beta I)^{-1}$ we can write $X = C(A + \beta I)$ and $y = Cb$.

3.4 ℓ_2 and ℓ_1 penalization (L_{21})

Finally, we can also consider the same nested optimization problem as in in the previous scheme, but with an ℓ_1 penalty used in the fixed-point operator, i.e.

$$\begin{aligned} u^* &= \arg\min_{u \in \mathbb{R}^k} \|\Phi u - (R + \gamma \Phi' w)\|_2^2 + \beta \|u\|_2^2 \\ w^* &= \arg\min_{w \in \mathbb{R}^k} \|\Phi w - \Phi u^*\|_2^2 + \beta' \|w\|_1 \end{aligned} \tag{11}$$

Here we can again use the simplification from (10) to write the solution as

$$w^* = \arg\min_{w \in \mathbb{R}^k} \| \underbrace{(\Phi - \gamma \Sigma \Phi')}_{X} w - \underbrace{\Sigma R}_{y} \|_2^2 + \beta' \|w\|_1. \tag{12}$$

As a result we have now transformed L_{21} into a standard Lasso problem, in terms of X and y, to which we can apply any off-the-shelf solution method.

4 Standardizing the data

In standard applications of the Lasso, it is often assumed that the feature matrix has columns that are standardized (i.e. that they are centered and zero-mean) and that the response vector is centered. Although we will briefly discuss the reasons for this, a more comprehensive treatment is given in e.g. [7, Section 3.4]. The centering is assumed because we generally want to estimate an unpenalized bias term w_0 so as to avoid making the problem dependent on the responses' origin. In doing so, the bias is given by the mean response and the remaining weights w can then be estimated using no bias term and centering the features and responses. The scaling of the features is perhaps more important and we can first note that the Lasso estimate is not invariant to this scaling. The scaling essentially evens the playing field for deciding which features are important, but it can also greatly impact the convergence speed of solution methods [7]. In this section we will now describe how to incorporate these assumptions into the L_{21} scheme introduced in the previous section.

We will now explicitly introduce a bias term w_0 into the value function approximation with $V^\pi(x) \approx \phi(x)^T w + w_0$. We can then rewrite the nested optimization problem as

$$(u^*, u_0^*) = \arg\min_{u,u_0} \|\Phi u + u_0 - (R + \gamma(\Phi' w + w_0))\|_2^2 + \beta_2 \|u\|_2^2 \tag{13}$$

$$(w^*, w_0^*) = \arg\min_{w,w_0} \|\Phi w + w_0 - (\Phi u^* + u_0^*)\|_2^2 + \beta_1 \|w\|_1. \tag{14}$$

Before solving these problems we will first introduce the following notation: $\overline{\Phi} = \text{mean}(\Phi)$ is the row-vector of feature means, $\widetilde{\Phi} = \Phi - \mathbf{1}_n \overline{\Phi}$ are the centered features where $\mathbf{1}_n$ is a column vector of n ones, and $\widehat{\Phi} = \widetilde{\Phi}\Omega$ consists of the centered and rescaled features given a scaling matrix Ω whose diagonal elements consist of the inverse standard deviations of the feature matrix Φ. Similar terms are introduced for centering both Φ' and R.

For both the projection and fixed-point sub-problems we can solve for the bias and weight terms invididually. In both cases the bias is given by the mean response minus the mean of the features and the optimal weights. For the bias of the projection step this can be written as

$$u_0^* = (\overline{R} + \gamma \overline{\Phi}' w + \gamma w_0) - \overline{\Phi} u^*. \tag{15}$$

The bias now depends upon finding u^*, but we can solve for this by centering both the features and responses and solving the following problem:

$$\Omega^{-1} u^* = \arg\min_{u \in \mathbb{R}^k} \|\widehat{\Phi} u - (\widetilde{R} + \gamma \widetilde{\Phi}' w)\|_2^2 + \beta_2 \|u\|_2^2 \tag{16}$$

$$= (\widehat{\Phi}^T \widehat{\Phi} + \beta_2)^{-1} \widehat{\Phi}^T (\widetilde{R} + \gamma \widetilde{\Phi}' w). \tag{17}$$

Note, however, that since we are solving this minimization problem using a scaled feature matrix (i.e. $\widehat{\Phi}$) we must remember to rescale back into the original units, which accounts for the the inverse of Ω.

We can now write the projected value function as

$$\begin{aligned}\Phi u^* + u_0^* &= (\Phi - \mathbf{1}_n\overline{\Phi})u^* + (\overline{R} + \gamma\overline{\Phi}'w + \gamma w_0) \\ &= \underbrace{\widehat{\Phi}(\widehat{\Phi}^T\widehat{\Phi} + \beta_2)^{-1}\widehat{\Phi}^T}_{\Sigma}(\widetilde{R} + \gamma\widetilde{\Phi}'w) + (\overline{R} + \gamma\overline{\Phi}'w + \gamma w_0).\end{aligned}$$

Here we have combined terms using the definitions introduced earlier and we can see the Σ term is the ℓ_2 penalized projection matrix onto the *centered and rescaled* features. Plugging this projected value function into the fixed-point problem we arrive at

$$\begin{aligned}(w^*, w_0^*) &= \arg\min_{w,w_0} \|\Phi w + w_0 - \Phi u^* - u_0\|_2^2 + \beta_1\|w\|_1 \\ &= \arg\min_{w,w_0} \| \underbrace{(\Phi - \gamma\Sigma\widetilde{\Phi}' - \gamma\mathbf{1}_n\overline{\Phi}')}_{X} w + (1-\gamma)w_0 - \underbrace{(\Sigma\widetilde{R} + \overline{R})}_{y} \|_2^2 + \beta_1\|w\|_1.\end{aligned}$$

We can see then that this is very closely related to the original formulation from (12), however here we are projecting according to the centered/rescaled features and X and y have additional components related to the mean next-state features and mean rewards respectively.

Now we truly have a standard Lasso problem. We can solve for the optimum w^* using any off-the-shelf Lasso solver using the scaled/centered matrix X and centered vector y (and again if we rescale X we must return the output to the original scaling). Given the optimal weights we can then solve for the bias term as $(1-\gamma)w_0^* = \overline{y} - \overline{X}w^*$, where these terms again denote the mean response and the mean of the features respectively.

5 Discussion of the different regularization schemes

In this work, we are particularly interested in the situation where the number of features is greater than the number of samples, $k > n$. Drawing on results from standard regression problems, we would expect the ℓ_2-based methods to perform poorly in this situation. In fact, we will see in the later experiments that just such a drop-off in performance occurs at the point when k overtakes n. We would, however, expect the ℓ_1-based methods, due to the feature selection properties of this penalty, to continue performing well so long as there is some small subset of relevant features that fit the value function well. In the setting of regression this behavior is well-established both empirically and theoretically [14, 3]. The extension of this behavior to the RL setting, although expected, is not entirely straight-forward due to the fixed-point aspect of the learning process. In the rest of this section, we will discuss and contrast how the ℓ_1 regularization is implemented in the two penalized methods, L_1 and L_{21}.

The main difference between the two schemes, L_1 and L_{21}, lies in the choice of where to place the ℓ_1 penalty. The L_1 approach uses a penalty directly in the projection operator. This is a straight-forward modification of the projection, and we would expect that, if the result of applying the Bellman operator can be well-represented by a sparse subset of features, the projection will find this. In fact, there are some recent theoretical guarantees that, if the target value

function V^π is sparse, the L_1 scheme will be able to take advantage of this fact [8]. More precisely, if $s \ll k$ is the number of features needed to represent V^π, the prediction error of L_1 is shown to directly scale with s instead of k [8]. This suggests that L_1 will perform well until the number of samples is bigger than the number of relevant features, $n \geq s$. However, we must note that when the projection step is combined with the fixed point, it is not entirely clear what is being optimized by the L_1 procedure. In fact, in [10] it is claimed that this approach does not correspond to any optimization problem in w. Further, from an algorithmic point of view, since the fixed-point is applied *after* the ℓ_1 optimization, this approach does not correspond to a Lasso problem, resulting in the need to use more specialized algorithms to solve for w.

Alternatively, the L_{21} approach places an ℓ_1 penalty in the fixed-point step. The main benefit of this approach is that it allows us to cast the regularized LSTD problem as a Lasso problem, to which we can apply general-purpose Lasso or ℓ_1 solvers. This also allows us to more straightforwardly apply results such as those of Section 4, something that is not easily done in the L_1 scheme. The application of standardization to the feature matrix has a great deal of impact on the results of Lasso in regression problems and we would expect it to have a great deal of impact for RL as well (in the later experiments we will see some evidence of this). We also found that the ability to standardize the features played a role in the previously mentioned flexibility of L_{21}. In fact, we found that without standardization of the features it was difficult to apply certain iterative methods such as those discussed in [12, 6] due to slow convergence.

One potential downside to the L_{21} approach is the necessity of using an ℓ_2 penalty in the projection step. It is not entirely clear what effect this has on the resulting algorithm, although in our preliminary experiments we found that this penalty was primarily useful in computing the projection matrix Σ, and making sure that the required matrix was non-singular. Further, the necessity of computing Σ does make L_{21} somewhat more expensive than L_1, however as this matrix only depends on Φ, using this procedure inside of a policy iteration scheme would only require the matrix being computed once. Finally, while there does exist some theoretical evidence that the L_1 approach is able to take advantage of the sparsity of V^π, no such guarantees exist yet for L_{21}. Our experiments, however, seem to indicate that L_{21} is able to capitalize on such value functions, and may even perform better when the value function is "truly sparse".

6 Experimental results

In comparing the performance of the regularization schemes introduced in Section 3, we will consider the *chain problem* introduced in [11]. In these experiments we will utilize a 20-state, 2-action MDP wherein states are connected in a chain and where upon taking action *left* from state x the system transitions to state $x-1$ with probability p and $x+1$ with probability $1-p$. The same holds for action *right* but in reverse and at both ends of the chain a successful right (or left) action leaves the state unchanged. We will restrict ourselves to the problem of policy evaluation and will evaluate the performance of the regularization schemes as the relative number of features k (as compared to the number of samples n) is varied. In particular, we will consider $k = s + \overline{s}$ features consisting

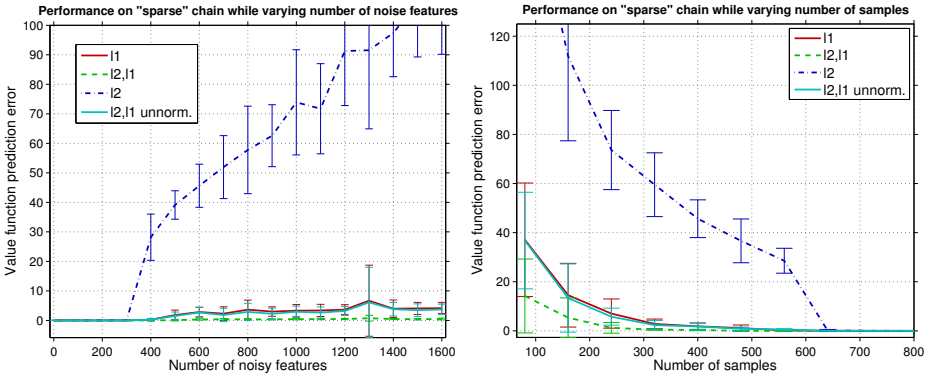

Fig. 2. Performance of policy evaluation on a sparse value function.

of s "relevant" features including a single constant feature and some number of radial-basis functions (RBFs) spread uniformly over the state space. The additional features consist of "irrelevant" or "noise" features implemented by extending the state-space of the chain model to include $\overline{s}$ additional dimensions where each additional dimension is independently distributed $x_t^{i+1} \sim \mathcal{N}(0, \sigma^2)$ for all time indices t. The value of each noise feature is then given by the corresponding state's value, i.e. $\phi_{s+i}(x_t) = x_t^{i+1}$ for $1 \leq i \leq \overline{s}$. These additional features can also be thought of as *random* features of the standard chain model.

We will first consider the case where the value function can be constructed as a sparse combination of features. In order to test this scenario, we start with a sparse value function and work backwards. We will consider a value function given as a linear combination of s relevant features, i.e. $V(x) = \sum_{i=1}^{s} \phi_i(x) w_i^*$ and we then define the reward of our model as $r(x, x') = V(x) - \gamma V(x')$. We have, in this case, artificially constructed our reward in order to enforce the desired sparse value function. In this experiment we used $s = 10$ and varied the number irrelevant features. In each of 20 runs we sample n samples on-policy and use them to approximate the value function. Here we used a policy π which takes action *left* on the first 10 states and action *right* on the next 10, although due to our construction of the reward function for this problem, the actual policy used does not affect the value function. The resulting approximation was then evaluated at 500 test points and the empirical error between this and the true value function is computed. The policy is kept fixed across all runs and within each run we perform cross-validation over the regularization parameter.

In Figure 2 we compare the L_2, L_1, and L_{21} variants described in Section 3; here we omit the L_{22} variant to avoid clutter[3]. L_{21} can be run using any Lasso solver. In our implementation we both tested LARS [4] and a number of gradient descent procedures [12, 6]. In order to generate the value function we sampled our true weights w_i^* uniformly in the range $[-5, 5]$ for every run. From our experiments, however, we saw that the true values of w did not play a significant role in explaining the error of each method: this variance was more attributable

[3] Preliminary experimental results seemed to show that the L_{22} variant performed similarly to the L_2 scheme.

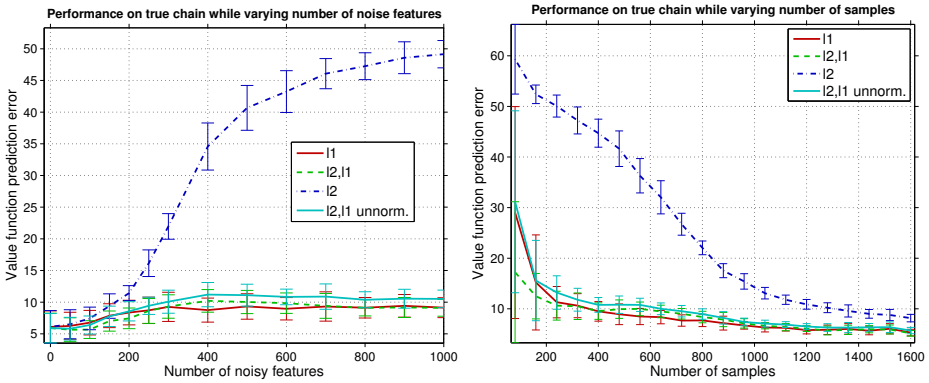

Fig. 3. Performance of policy evaluation on the chain model for a fixed policy.

to the data samples generated. The first of these plots shows the behavior using $n = 400$ samples (consisting of 20 episodes of length 20) while varying the number of noise features. We can first note that all algorithms perform quite well until the number of features k approaches the number of samples n, but after around 400 noise features the algorithms begin to differentiate. L_2 immediately starts overfitting the data and its prediction error grows almost linearly with the total number of features k. On the other hand, the ℓ_1 regularization approaches are expected to depend more directly on the *true* dimensionality s rather than k. In fact, we can see that both L_1 and L_{21} are not drastically affected by overfitting when k becomes greater than n. In the plot, we also include the performance of L_{21} without the rescaling described in Section 4 and here this performs about as well as the L_1 approach (using LARSTD). The rescaled version, however, is much better able to deal with the increase in number of noise features as can be seen by its very low prediction error, and in comparison to the other methods this is nearly constant. The second plot shows the results of $\overline{s} = 600$ noise features and varying the number of samples n, and we see similar differences in performance.

Next we consider the standard chain problem which uses a reward of 1 when the relevant state (x_1) is at either endpoint and 0 elsewhere. In the first of these experiments we use $s = 6$ relevant features and again vary the number irrelevant features. In each of 20 runs we again sample 400 data points (consisting of 20 episodes of length 20) on-policy and use these points to approximate the value function. Given this model we can compute the true value function and in order to generate the plots we evaluate the empirical error between this and the estimated value function at 500 test points. Again, the policy is kept fixed and within each run we perform cross-validation over the regularization parameter.

The results of Figure 3 closely echo the results of Figure 2. Here, however, we can see that there is a somewhat more significant difference between the unscaled version of L_{21} and the L_1 variant, although this performance is still significantly better than L_2 alone. We also see that the prediction error is no longer zero (as would be expected due to the fact that now the value function can be exactly represented in the linear space). We can also see, however, that there is a more significant increase in this error as we approach the point where the number of

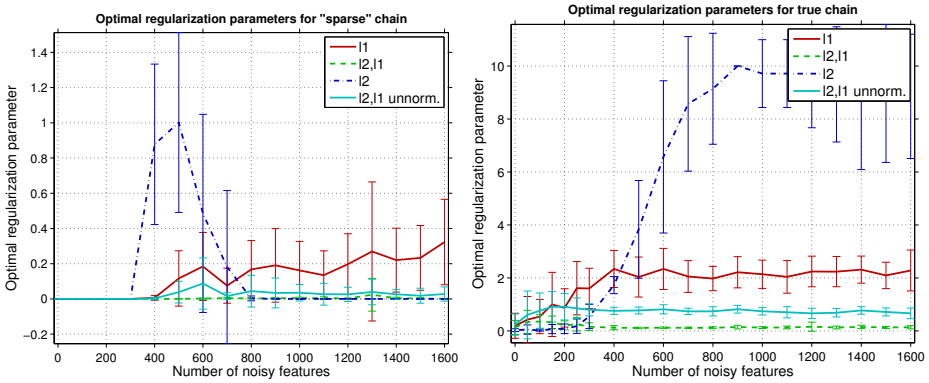

Fig. 4. Effect of the number of irrelevant features on the optimal penalty parameters chosen via cross-validation.

features equals the number of samples. Again, the second plot shows the results of $\overline{s} = 600$ noise features while varying the number of samples n, and again we see similar differences in performance.

In Figure 4 we show the effect of the number of irrelevant features on the best penalty parameters chosen via cross-validation on both the sparse chain and the standard chain model. The ℓ_1 based methods use LARS (or the LARSTD modification) to find the regularization path and for the ℓ_2 based method we used a grid of 20 parameters logarithmically spaced between 10^{-6} and 10. In both figures we show results where the number of samples is fixed at $n = 400$ and the number of noisy features is increased. The main point to see here is the much greater variability in this parameter value for the L_2 method. The L_1 method then may also be more variable than the L_{21} method, but this level of increase could also be attributable to different scaling in this parameter. For these experiments we also found that the ℓ_2 parameter of L_{21} was primarily useful in ensuring the relevant matrix Σ was nonsingular, and as a result we were able to set this parameter to a quite low-value 10^{-6} and in these plots we only show the ℓ_1 penalty. Based on this it would seem that the tuning of the penalty parameter for L_{21} is at least *no harder* than that of L_1 due to the minimal effect of the second parameter Finally, we point out that the decline in the penalty parameter of L_2 for the sparse chain is attributable to the fact that after approximately 600–800 noise features all parameters were "equally bad".

Finally, we analyzed the effect of the number irrelevant features on the number of iterations when solving the problem using L_1/LARSTD and when solving the L_{21} problem using LARS [4]. Although these results are not entirely conclusive, it appears that the L_{21} approach may require fewer iterations for sparse models, but more for non-sparse models. As mentioned earlier, the additional flexibility of the L_{21} scheme allows us to apply more sophisticated optimizations schemes such as the coordinate descent approach of [6] or other iterative schemes [12]. We applied both of these procedures, code for which is available online and is quite fast, to our problem. However, the comparison of the complexity of these different algorithms is not simple and we leave it for future work.

7 Conclusion

In this paper we presented a number of regularization schemes for LSTD, culminating in a novel approach using nested ℓ_2 and ℓ_1 penalties. We then showed how to apply standardization techniques from the regression literature to this problem and discussed preliminary experiments comparing this approach to other schemes based on ℓ_2 and ℓ_1 penalized projection. The main benefit of this approach is the additional flexibility it provides over ℓ_1 projected LSTD approaches (such as LARSTD), and the ability to use standard Lasso solvers and techniques. This means that any new solver for Lasso could be immediately used in L_{21}, whereas this would need a rethinking of the whole algorithm for L_1.

There are also a number of open areas of research with this approach. A more thorough experimental analysis is needed, including an extension of this method to the problem of policy iteration via LSPI. A theoretical analysis of this approach is also called for, in line with the previously noted analysis of the L_1 scheme presented in [8]. Finally, a better understanding of the ramifications of including the ℓ_2 penalty would be most beneficial.

References

1. Antos, A., Szepesvári, C., Munos, R.: Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. Machine Learning 71(1) (2008)
2. Bradtke, S., Barto, A.: Linear least-squares algorithms for temporal difference learning. Machine Learning 22, 33–57 (1996)
3. Bunea, F., Tsybakov, A., Wegkamp, M.: Sparsity oracle inequalities for the lasso. Electronic Journal of Statistics 1, 169–194 (2007)
4. Efron, B., Hastie, T., Johnstone, I., Tibshirani, R.: Least angle regression. Annals of statistics 32(2) (2004)
5. Farahmand, A., Ghavamzadeh, M., Szepesvari, C., Mannor, S.: Regularized policy iteration. Advances in Neural Information Processing Systems 21 (2009)
6. Friedman, J., Hastie, T., Höfling, H., Tibshirani, R.: Pathwise coordinate optimization. The Annals of Applied Statistics 1(2), 302–332 (2007)
7. Friedman, J., Hastie, T., Tibshirani, R.: The elements of statistical learning. Springer (2001)
8. Ghavamzadeh, M., Lazaric, A., Munos, R., Hoffman, M.: Finite-sample analysis of Lasso-TD. In: Proceedings of the International Conference on Machine Learning (2011)
9. Johns, J., Painter-Wakefield, C., Parr, R.: Linear complementarity for regularized policy evaluation and improvement. Advances in Neural Information Processing Systems 23 (2010)
10. Kolter, J., Ng, A.: Regularization and feature selection in least-squares temporal difference learning. In: Proceedings of the International Conference on Machine Learning (2009)
11. Lagoudakis, M.G., Parr, R.: Least-squares policy iteration. Journal of Machine Learning Research 4 (2003)
12. Schmidt, M.: Graphical Model Structure Learning with l1-Regularization. Ph.D. thesis, University of British Columbia (2010)
13. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. MIT Press (1998)
14. Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological) 58(1), 267–288 (1996)