Regularized Fitted Q-iteration: Application to Bounded Resource Planning

Amir massoud Farahmand[1], Mohammad Ghavamzadeh[1], Csaba Szepesvári[1], and Shie Mannor[2]

[1] Department of Computing Science, University of Alberta,
Edmonton, AB T6G 2E8, Canada
{amir,mgh,szepesva}@cs.ualberta.ca
[2] Department of Electrical & Computer Engineering, McGill University,
Montreal, QC H3A 2A7, Canada
shie.mannor@mcgill.ca

Abstract. We consider bounded resource planning in a Markovian decision problem, i.e., the problem of finding a good policy given access to a generative model of the environment and a limit on the computational resources. We propose to use fitted Q-iteration algorithm with penalized (or regularized) least-squares regression as the regression subroutine to address the problem of selecting an appropriate function approximator in each iteration. The algorithm is presented in detail for the case when the function space is a reproducing-kernel Hilbert space underlying a user-chosen kernel function. We derive bounds on the quality of the solution and argue that how data-dependent penalties can lead to almost optimal performance. A simple example is used to illustrate the benefits of using a penalized procedure.

1 Introduction

Regularization has proven an effective tool in machine learning and in particular in supervised learning. The main idea is to consider the learning problem as an optimization problem where one minimizes the sum of an empirical error term and a complexity penalty (regularizer) that penalizes complex solutions. The tradeoff between the empirical error term and the penalty term is controlled by a single numerical value, the regularization coefficient, which multiplies the penalty term. This way the problem of model selection in a given family of functions is reduced to the problem of choosing a single numerical value. When the parameter is chosen in an appropriate way based on the data or by complexity regularization, the resulting procedure is known to adapt to the complexity of the target function automatically, converging almost as fast as if the model was known beforehand (e.g., [9]).

Recently the problem of tuning function approximators has received considerable attention in the reinforcement learning (RL) community. For example, [15] considered parameterized function approximation architecture where the parameters are changed to better minimize the Bellman residual error, and [19] constructed new basis functions from the Bellman residual in fitted value iteration. (For other similar approaches see the references in these papers.) Furthermore, non-parametric regression methods are also used where the function representation can potentially adapt to the actual difficulty of the problem. Examples of this approach include the kernel-based RL method [18] that uses kernel-regression, the work of [12] where support vector regressors are used to represent policies in an approximate policy iteration procedure, the regression tree-based fitted Q-iteration algorithm [7], or the GPTD algorithm [6] that builds on Gaussian processes regression. Moreover, nonparametric regression-based approaches have recently been used in computational finance in the closely related problem of pricing financial derivatives [22, 13].

In this work we consider a nonparametric regression based approach, but one that builds on penalized least-squares. Given that penalized least-squares is one of the most successful approaches to supervised regression, it is surprising that it has not been thoroughly investigated in RL. The only works that we know of that used this approach are [10], [14] and [8]. Penalized least-squares with L^2-penalties for finding the value function of a policy given a trajectory in a deterministic system were explored by [10], while L^1-penalties for the same problem were considered in [14]. Motivated by the fact that the straightforward implementation of penalized least-squares involves a nontrivial computational cost, both papers focused on computational efficiency. On the other hand, [8] recently analyzed Regularized Policy Iteration methods that use Least Squares Temporal Difference (LSTD) and a modified version Bellman Residual Minimization (BRM) for policy evaluation and provided convergence results.

The problem considered in this paper is planning with bounded computational resources in a discounted Markovian Decision Problem (MDP): this is the problem of finding a good policy given a limit on the amount of computation. We do not assume the availability of a full model, but only that we can generate transitions for any given action at any selected state. Such generative models have been explored by a number of works (e.g. [11, 17]). Our work is complementary: We are guaranteed to achieve optimality in the limit (unlike [17]) and we do not scale exponentially with the effective planning horizon (unlike [11]). However, our method comes with other restrictions: The value functions should come from "nice" function spaces.

In this paper, we consider the fitted Q-iteration algorithm of [7] where the iterates are obtained by solving penalized least-squares regression problems. This way we hope to borrow the strength of a state-of-the-art supervised learning approach to help solving planning problems more efficiently. We develop specific formulae for kernel-based fitted Q-iteration. Our main theoretical results bound the quality of the solutions given that the algorithm spends a finite amount of computational resources on the task. The strength of the approach is that the complexity of the function class (and thus the performance) can be controlled by tuning the penalty factor alone. We argue that non-trivial performance gains are possible if one chooses the regularization coefficient in a data-dependent manner.

Although finite-sample performance of a generic fitted Q-iteration has been considered earlier [2, 1], to the best of our knowledge, this is the first work that addresses finite-sample performance of a *regularized* RL algorithm and gives a concrete algorithm to implement. The analysis presented here builds on these previous works, but extends and improves them. The improvement is possible partially because of the differences in the problem setups, e.g. in the previous works only a single trajectory was available while here we assume the availability of a generative model. The differences will be further explained after introducing the results.

The rest of the paper is organized as follows. In Section 2, we present the notations of infinite space MDP which is used in the paper. In Section 3, we recall the fitted Q-iteration algorithm, which is the main algorithm studied in this paper. The main result that relates the error during iterations with the eventual value function L^p norm error is presented in Section 4. Bounds for L^2 regularization are given in Section 5. Experimental illustration of the algorithm is presented in Section 6.

2 Background and notation

Because we consider continuous state spaces, we need a few concepts from analysis. These are introduced first. This is followed by the introduction to the Markovian Decision Problems (MDPs). We refer the reader to [3] for further details.

For a measurable space with domain S, we let $\mathcal{M}(S)$ denote the set of probability measures over S. For $p \geq 1$, a measure $\nu \in \mathcal{M}(S)$, and a measurable function $f : S \to \mathbb{R}$, we let $\|f\|_{p,\nu}$ denote the $L^p(\nu)$-norm of f:

$$\|f\|_{p,\nu}^p = \int |f(s)|^p \nu(ds).$$

We shall also write $\|f\|_\nu$ to denote the $L^2(\nu)$-norm of f. We denote the space of bounded measurable functions with domain $\mathcal{X}$ by $B(\mathcal{X})$, and the space of measurable functions with bound $0 < K < \infty$ by $B(\mathcal{X}; K)$.

A finite-action discounted MDP is defined by a quintuple $(\mathcal{X}, \mathcal{A}, P, S, \gamma)$, where $\mathcal{X}$ is the (possibly infinite) *state space*, $\mathcal{A} = \{a_1, a_2, \ldots, a_M\}$ is the finite set of *actions*, $P : \mathcal{X} \times \mathcal{A} \to \mathcal{M}(\mathcal{X})$ is the *transition probability kernel*, $P(\cdot|x, a)$ defining the next-state distribution upon taking action a in state x, $S(\cdot|x, a)$ gives the corresponding distribution of *immediate rewards*, and $\gamma \in (0, 1)$ is the discount factor. We make the following assumptions on the MDP:

Assumption A1 (MDP Regularity) $\mathcal{X}$ is a compact subset of the d-dimensional Euclidean space. We assume that the random immediate rewards are bounded by $\hat{R}_{\max}$ and the expected immediate rewards $r(x, a) = \int r S(dr|x, a)$ are bounded by $R_{\max}$: $\|r\|_\infty \leq R_{\max}$. (Note that $R_{\max} \leq \hat{R}_{\max}$.)

A stationary Markov policy $\pi : \mathcal{X} \to \mathcal{M}(\mathcal{A})$ is defined as a time-independent (measurable) mapping from the current state x to a distribution over the set of actions $\pi(\cdot|x)$. A policy is deterministic if the probability distribution concentrates on a single action for all states. Deterministic stationary Markov policies will be identified with mappings from states to actions $\pi : \mathcal{X} \to \mathcal{A}$. In the rest of this paper, we use the term policy to refer to stationary Markov policies.

The value of a policy π when it is started from a state x is defined as the total expected discounted reward that is encountered while the policy is executed:

$$V^\pi(x) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_t \middle| X_0 = x \right].$$

Here R_t denotes the reward received at time step t; $R_t \sim S(\cdot|X_t, A_t)$ and X_t evolves according to $X_{t+1} \sim P(\cdot|X_t, A_t)$ where A_t is sampled from the distribution assigned to the past observations by π. For a policy π, $A_t \sim \pi(\cdot|X_t)$, and if π is deterministic we write $A_t = \pi(X_t)$. The function V^π is also called the state-value function of policy π. Closely related to the state-value functions are the action-value functions, defined by

$$Q^\pi(x, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_t \middle| X_0 = x, A_0 = a \right].$$

In words, the action-value function underlying π, denoted by $Q^\pi(x, a)$, is the expected discounted return when the decision process is started in state x and the first action is a while all the subsequent actions are determined by the policy π. It is easy to see that for any policy π, the functions V^π and Q^π are bounded by $R_{\max}/(1-\gamma)$.

Given an MDP, the goal is to find a policy that attains the best possible values,

$$V^*(x) = \sup_\pi V^\pi(x),$$

for all states $x \in \mathcal{X}$. Function V^* is called the optimal value function. A policy is called optimal if it attains the optimal values $V^*(x)$ for *any* state $x \in \mathcal{X}$, i.e., if $V^\pi(x) = V^*(x)$ for all $x \in \mathcal{X}$.

In order to characterize optimal policies it will be useful to define the optimal action-value function, $Q^*(x, a)$:

$$Q^*(x, a) = \sup_\pi Q^\pi(x, a).$$

Further, we say that a deterministic policy π is *greedy* w.r.t. an action-value function $Q \in B(\mathcal{X} \times \mathcal{A})$ and write $\pi = \hat{\pi}(\cdot; Q)$, if, for all $x \in \mathcal{X}$ and $a \in \mathcal{A}$, $\pi(x) \in \operatorname{argmax}_{a \in \mathcal{A}} Q(x, a)$.

Because $\mathcal{A}$ is finite, a greedy policy always exists no matter how Q is chosen. Greedy policies are important because any greedy policy w.r.t. Q^* is optimal. Hence, to find an optimal policy it suffices to determine Q^* and the search for optimal policies can be restricted to deterministic stationary Markov policies.

The Bellman optimality operator $T : B(\mathcal{X} \times \mathcal{A}) \to B(\mathcal{X} \times \mathcal{A})$ is defined by

$$(TQ)(x, a) = r(x, a) + \gamma \int \max_{a' \in \mathcal{A}} Q(y, a') P(dy|x, a).$$

```
**FittedQ**($D$,$K$,$Q_0$)
// $D$: samples
// $K$: number of iterations
// $Q_0$: Initial action-value function
$Q \leftarrow Q_0$ // Initialization
for $k = 0$ to $K - 1$ do
    $Q' \leftarrow \mathrm{FitQ}(Q, D, k)$
    $Q \leftarrow Q'$
end for
return $Q$
```

Fig. 1. Fitted Q-Iteration

As it is well known, this operator T is a contraction operator w.r.t. the supremum-norm with index γ. Moreover, the optimal action-value function is the unique fixed point of T: $TQ^* = Q^*$.

Throughout the paper $\mathcal{F} \subset \{ f : \mathcal{X} \to \mathbb{R} \}$ will denote some subset of real-valued functions over the state-space $\mathcal{X}$. For convenience, we will treat elements of $\mathcal{F}^M$ as real-valued functions f defined over $\mathcal{X} \times \mathcal{A}$ with the obvious identification $f \equiv (f_1, \ldots, f_M)$, $f(x, a_j) = f_j(x)$, $j = 1, \ldots, M$. The set $\mathcal{F}^M$ will denote the set of admissible functions used in the optimization step of our algorithm.

3 Algorithm

The algorithm studied in this paper is an instance of the generic fitted Q-iteration method, whose pseudo-code is shown in Fig. 1. The algorithm attempts to approximate the optimal action-value function Q^* and mimics value iteration. Since computing the Bellman operator applied to the last iterate at any point involves evaluating a high-dimensional integral, we use a Monte-Carlo approximation together with a regression procedure. For this purpose a set of samples, D is generated: $D = \{(X_1, A_1, R_1, X'_1), \ldots, (X_N, A_N, R_N, X'_N)\}$. For the sake of simplifying the analysis, we assume that the actions and next states are generated by some fixed stochastic stationary policy π_b: $A_t \sim \pi_b(\cdot|X_t)$, $X'_t \sim P(\cdot|X_t, A_t)$, $R_t \sim S(\cdot|X_t, A_t)$. Further, we assume that $\{X_t\}$ is an i.i.d. sequence and $(X_t, A_t) \sim \nu$. The state-marginal of ν is denoted by $\nu_{\mathcal{X}}$. We assume that ν is a strictly positive measure, i.e., its support is $\mathcal{X} \times \mathcal{A}$. Intuitively, this ensures that the samples cover all state-action pairs. In particular for this we must have that $\pi_{b0} \overset{\mathrm{def}}{=} \min_{a \in \mathcal{A}} \inf_{x \in \mathcal{X}} \pi_b(a|x) > 0$.

The fitting procedure that we study in this paper is penalized (regularized) least-squares. Assuming that in the k^{th} iteration we use samples with index

$N_k \le i < N_k + M_k = N_{k+1} - 1$, the $(k+1)^{\text{th}}$ iterate is obtained by

$$Q_{k+1} = \operatorname*{argmin}_{Q \in \mathcal{F}^M} \frac{1}{M_k} \sum_{i=N_k}^{N_k+M_k-1} \left[R_i + \gamma \max_{a' \in \mathcal{A}} Q_k(X'_i, a') - Q(X_i, A_i)\right]^2 + \lambda \mathrm{Pen}(Q), \tag{1}$$

where $\mathrm{Pen}(Q)$ is a customary penalty term and $\lambda > 0$ is the regularization coefficient.[3] The first term is the sample-based least-squares error of using $Q(X_i, A_i)$ to predict $R_i + \gamma \max_{a' \in \mathcal{A}} Q_k(X'_i, a')$ at (X_i, A_i). This term is the empirical counterpart to the loss $L_k(Q) = \mathbb{E}\left[(R_i + \gamma \max_{a' \in \mathcal{A}} Q_k(X', a') - Q(X, A))^2\right]$. The minimizer of this loss function is the regression function

$$\mathbb{E}\left[R_i + \gamma \max_{a' \in \mathcal{A}} Q_k(X'_i, a') \,|\, X_i = x, A_i = a\right] = (TQ_k)(x, a).$$

As the number of samples grows to infinite, the empirical loss converges to L_k and we expect the iterate Q_{k+1} to converge to TQ_k. To achieve this, one needs to balance between the expressiveness of the function class and its complexity (or the resulted function would be overfitted or underfitted). This is the job of the second term on the right hand side of (1). This term implicitly regulates how complex solutions are acceptable. Choosing a larger λ means searching in a smaller space of functions.

When $\mathcal{F}^M$ is a Sobolev-space[4] and $\mathrm{Pen}(Q)$ is the corresponding Sobolev-space norm (the squared norm of the generalized partials of Q), this optimization leads to thin-plate spline estimates, popular in the nonparametric statistics literature [9]. When searching for a solution in general the order of smoothness is unknown. Further, the optimal choice of the regularization coefficient would depend on the target function. The approach taken in regression can be followed here, too: Try different smoothness orders (this corresponds to different penalty terms) with different regularization coefficients and choose between them using a hold-out set. This leads to estimates whose rate of convergence has the optimal order and scales with the actual roughness, $\mathrm{Pen}(TQ_k)$.

Optimizing over a Sobolev-space is a particular case of optimization in a reproducing kernel Hilbert space (RKHS). Thus, more generally, we may start with a Mercer kernel function k and set $\mathrm{Pen}(Q)$ to be the norm of Q in $\mathcal{H}$, the RKHS underlying k [20]. This way we obtain

$$Q_{k+1} = \operatorname*{argmin}_{Q \in \mathcal{H}} \frac{1}{M_k} \sum_{i=N_k}^{N_k+M_k-1} \left[R_i + \gamma \max_{a' \in \mathcal{A}} Q_k(X'_i, a') - Q(X_i, A_i)\right]^2 + \lambda \left\|Q\right\|^2_{\mathcal{H}}. \tag{2}$$

[3] Note that in practice one would generate the samples whenever they are needed, i.e., there is no need to generate and store all the samples. However, it is also possible to reuse the samples if sample generation is expensive. In such a case the analysis needs to be changed slightly.

[4] Sobolev-spaces generalize Hölder spaces by allowing functions which are only almost everywhere differentiable. Thus, they can be useful for control problems where value-functions often have ridges.

According to the Representer Theorem (e.g., see [20]), every solution to Eq. (2) is the sum of kernels centered on the observed samples: i.e.,

$$Q(x,a) = \sum_{i=N_K}^{N_k+M_k-1} \alpha_{i-N_k+1} \mathrm{k}\big((X_i, A_i), (x,a)\big),$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{M_k})^\top$ are the coefficient that must be determined. Let us assume that Q_k was obtained previously in a similar form:

$$Q_k(x,a) = \sum_{i=N_{k-1}}^{N_{k-1}+M_{k-1}} \alpha^{(k)}_{i-N_{k-1}+1} \mathrm{k}\big((X_i, A_i), (x,a)\big),$$

and let us collect the coefficients into a vector $\boldsymbol{\alpha}^{(k)} \in \mathbb{R}^{M_{k-1}}$. Replacing Q in Eq. (2) by its expansion and using RKHS properties, we get

$$\boldsymbol{\alpha}^{(k+1)} = \operatorname*{argmin}_{\boldsymbol{\alpha} \in \mathbb{R}^{M_k}} \frac{1}{M_k} \left\| \boldsymbol{r} + \gamma \boldsymbol{K}^+ \boldsymbol{\alpha}^{(k)} - \boldsymbol{K}\boldsymbol{\alpha} \right\|^2 + \lambda \boldsymbol{\alpha}^\top \boldsymbol{K} \boldsymbol{\alpha}, \tag{3}$$

with $\boldsymbol{K} \in \mathbb{R}^{M_k \times M_k}$, $\boldsymbol{K}^+ \in \mathbb{R}^{M_k \times M_{k-1}}$,

$$\begin{aligned} [\boldsymbol{K}]_{ij} &= \mathrm{k}\big((X_{i-1+N_k}, A_{i-1+N_k}), (X_{j-1+N_k}, A_{j-1+N_k})\big), \\ [\boldsymbol{K}^+]_{ij} &= \mathrm{k}\big((X'_{i-1+N_k}, A^{(k)}_{i-1+N_k}), (X_{j-1+N_{k-1}}, A_{j-1+N_{k-1}})\big), \end{aligned}$$

where $A^{(k)}_j = \operatorname{argmax}_{a \in \mathcal{A}} Q_k(X'_j, a)$, and $\boldsymbol{r} = (R_{N_k}, \ldots, R_{N_k+M_k-1})^\top$. Solving Eq. (3) for $\boldsymbol{\alpha}$ we obtain

$$\boldsymbol{\alpha}^{(k+1)} = (\boldsymbol{K} + M_k \lambda \boldsymbol{I})^{-1} (\boldsymbol{r} + \gamma \boldsymbol{K}^+ \boldsymbol{\alpha}^k).$$

The computational complexity of iteration k with a straightforward implementation is $O(M_k^3)$ as it involves the inversion of a matrix. Thus, in order to understand how the algorithm behaves it suffices to understand how the error behaves after a certain number of iterations. This is what we do in the next two sections.

4 Error propagation

In order to analyze Fitted Q-iteration it is customary to rewrite it in the form

$$\begin{aligned} Q_{k+1} &= TQ_k - \varepsilon_k, \quad k \geq 0, \\ \varepsilon_{-1} &= Q^* - Q_0. \end{aligned} \tag{4}$$

Note that these equations define the error sequence ε_k ($\varepsilon_k : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$) from the sequence of iterates $\{Q_k\}$ and not vice versa (except for ε_{-1}, the "initial error", which is introduced with a unified notation so that some expressions below can be simplified). Here we are interested in studying how the errors $\{\varepsilon_k\}$ influence the performance of the policy greedy w.r.t. Q_K ($K > 0$ is the number of iterations in the algorithm; see Fig. 1). The idea is that the regression procedure controls the size of the error functions ε_k, hence it must be possible to obtain good policies eventually. For $k \geq 0$ let π_k be the greedy policy w.r.t. Q_k: $\pi_k = \hat{\pi}(\cdot; Q_k)$. Then our goal is to bound the norm of $V^* - V^{\pi_K}$ (by the definition of an optimal value function, this is guaranteed to be non-negative). In order to arrive at a bound on this quantity we make a number of assumptions.

Remember that ν denotes the distribution underlying $\{(X_t, A_t)\}$. For the sake of flexibility, we allow the user to choose another distribution, $\rho \in \mathcal{M}(\mathcal{X})$, to be used in assessing the procedure's performance. It turns out that in the technique that we use to bound the final error as a function of the intermediate errors we need to change distributions between future state-distributions started from ρ and ν. A natural way to bound the effect of changing from measure α to measure β is to use the Radon-Nikodym derivative of α w.r.t. β:[5] for any nonnegative measurable function f, $\int f\, d\alpha = \int f \frac{d\alpha}{d\beta}\, d\beta \leq \|\frac{d\alpha}{d\beta}\|_\infty \int f\, d\beta$. This motivates the following definition, very similar to the one introduced by [16]:

Definition 1 (Discounted-average Concentrability of Future-State Distribution) *Given $\rho \in \mathcal{M}(\mathcal{X})$, $\nu \in \mathcal{M}(\mathcal{X} \times \mathcal{A})$, $m \geq 0$ and an arbitrary sequence of stationary policies $\{\pi_m\}_{m \geq 1}$ let $\rho^{\pi_1,\ldots,\pi_m} \in \mathcal{M}(\mathcal{X} \times \mathcal{A})$ denote the future state-action distribution obtained when the first state is obtained from ρ and then we follow policy π_1, then policy π_2, ..., then π_{m-1} at which step a random action is selected with π_m. Define*

$$c_{\rho,\nu}(m) = \sup_{\pi_1,\ldots,\pi_m} \left\| \frac{d(\rho^{\pi_1,\ldots,\pi_m})}{d\nu} \right\|_\infty ,$$

with the understanding that $c_{\rho,\nu}(m) = \infty$ if the future state-action distribution $\rho^{\pi_1,\ldots,\pi_m}$ is not absolutely continuous w.r.t. ν. The first-order k-shifted ($k \geq 0$, $k \in \mathbb{N}$) discounted-average concentrability of future-state distributions is defined by

$$C^{(1,k)}_{\rho,\nu} = (1-\gamma) \sum_{m=0}^{\infty} \gamma^m c_{\rho,\nu}(m+k).$$

Similarly, the second-order k-shifted ($k \geq 0$, $k \in \mathbb{N}$) discounted-average concentrability of future-state distributions is defined by

$$C^{(2,k)}_{\rho,\nu} = (1-\gamma)^2 \sum_{m \geq 1} m \gamma^{m-1} c_{\rho,\nu}(m+k).$$

In general $c_{\rho,\nu}(m)$ diverges to infinity as $m \to \infty$. However, thanks to discounting, $C^{(i,j)}_{\rho,\nu}$ will still be finite whenever γ^m converges to zero faster than $c_{\rho,\nu}(m)$ converges to ∞. In particular, if the rate of divergence of $c_{\rho,\nu}(m)$ is

[5] The Radon-Nikodym (RN) derivative is a generalization of the notion of probability densities. According to the Radon-Nikodym Theorem, $d\alpha/d\beta$, the RN derivative of α w.r.t. β is well-defined if β is σ-finite and if α is absolute continuous w.r.t. β. In our case β is a probability measure, so it is actually finite.

sub-exponential, i.e., if $\Gamma = \limsup_{m\to\infty} 1/m \log c_{\rho,\nu}(m) \leq 0$ then $C^{(i,j)}_{\rho,\nu}$ will be finite. Note that the definition given here is not identical to the previous similar definition by [16]. The main difference is that unlike in [16] here ρ is a distribution over the states and ν is a distribution over state-action pairs. The reason is that here we work with action-value functions, while [16] considered state-value functions. Note that it is possible to avoid changing this definition (as it was done in [1]), but the price is that the bounds will be more conservative. Interestingly, the bounds here avoid the supremum norm arguments used by [1] and are thus less conservative.

The main result of this section is the following theorem that bounds the loss of using the learned policy π_K as a function of the losses of the solutions of the regression problems solved while running the algorithm:

Theorem 1 (L^p-bound) *Consider a discounted MDP with a finite number of actions. Let $p \geq 1$. Assume that Q_k and ε_k satisfy* (4) *and that π_k is a policy greedy w.r.t. Q_k. Fix $K > 0$. Define $E_0 = \|\varepsilon_{-1}\|_\infty$ and $\overline{\varepsilon}_K = \max_{0\leq k\leq K} \|\varepsilon_k\|_{p,\nu}$. Then,*

$$\|V^* - V^{\pi_K}\|_{p,\rho} \leq 2\left[\frac{1}{1-\gamma} + \frac{\gamma}{(1-\gamma)^2}\right]\gamma^{K/p}E_0 + 2\left[\frac{(C^{(1,1)}_{\rho,\nu})^{1/p}}{1-\gamma} + \frac{\gamma\,(C^{(2,1)}_{\rho,\nu})^{1/p}}{(1-\gamma)^2}\right]\overline{\varepsilon}_K.$$

5 L^2-bound for regularized kernel-based regression

In this section we assume that Q_{k+1} is obtained by solving the RKHS regularization problem of Eq. (2). By using Prop. 3 of [23], the following generalization of Theorem 21.1 of [9] to arbitrarily RKHS with smooth kernel functions can be obtained. The result is for the case when $\mathcal{X} = [0,1]^d$, but can be generalized to other compact spaces with "regular" boundaries relatively easily.

Theorem 2 *Assume that $\mathcal{X} = [0,1]^d$, $\mathrm{k} \in \mathrm{Lip}^*(s, C(\mathcal{X},\mathcal{X}))$ and Q_k is such that $TQ_k \in \mathcal{H}(=\mathcal{H}_{\mathrm{k}})$.*[6] *Furthermore, (for the sake of simplicity) assume that all functions involved in the regression problem (the reward function, Q_k, and the result of the optimization problem(Q_{k+1}) are bounded by some constant $L > 0$.*[7] *Let Q_{k+1} be the solution of* (2) *with some $\lambda > 0$. Then*

$$\|Q_{k+1} - TQ_k\|^2_\nu \leq 2\lambda\,\|TQ_k\|^2_{\mathcal{H}} + \frac{c_1 L^4}{M_k \lambda^{d/s}} + \frac{c_2 \log(1/\delta)}{M_k L^4}$$

with probability at least $1-\delta$, for some $c_1, c_2 > 0$.

Note the trade-off in the bound: increasing λ increases the first term, but decreases the second. The optimal choice strikes a balance between these two terms. It depends on the number of samples M_k, the complexity of the target function

[6] For the definition of the generalized Lipschitz space Lip^* see [23].

[7] When this does not hold, a truncation argument is needed, but the result would essentially be left unchanged.

TQ_k measured by $\|TQ_k\|_{\mathcal{H}}^2$, the dimension of the problem d, and the degree of smoothness measured by s. With $\lambda = cM_k^{-1/(1+d/s)}$ the rate of convergence is $O(M_k^{-1/(1+d/s)})$, showing that smoother problems makes the problem easier - an intuitive result. To find the optimal λ in a data-dependent manner, one may use a hold-out set or any other model selection method. Note that the task of model selection is reduced to selecting a single number, i.e. regularization coefficient.

As an immediate corollary of this result and Theorem 1 we get the following result, assuming that in each iteration we are using the same regularization parameter.

Corollary 3 (L^2-bound) *Assume that the conditions of the previous theorem hold and we use the same number of samples in each iteration: $M_1 = M_2 = \ldots = M_K$. Let π_K be greedy w.r.t. the K^{th} iterate, Q_K. Define $E_0 = \|\varepsilon_{-1}\|_\infty$ and let $B = \max_{0\le k\le K} \left\|T^k Q_0\right\|_{\mathcal{H}}^2$. Then, for any $\delta > 0$ with probability at least $1-\delta$,*

$$\|V^* - V^{\pi_K}\|_\rho \le 2\left[\frac{1}{1-\gamma} + \frac{\gamma}{(1-\gamma)^2}\right]\gamma^{K/2}E_0 + \\ 2\left[\frac{(C_{\rho,\nu}^{(1,1)})^{1/2}}{1-\gamma} + \frac{\gamma (C_{\rho,\nu}^{(2,1)})^{1/2}}{(1-\gamma)^2}\right]\left[c_1\lambda B + \frac{c_2 L^4}{M_1 \lambda^{d/s}} + \frac{c_3 \log(K/\delta)}{M_1 L^4}\right]^{1/2}$$

for some universal constants $c_1, c_2, c_3 > 0$.

Note that by choosing $\lambda = cM_1^{-1/(1+d/s)}$ the second term is made converging to zero with $M_1 \to \infty$ at a rate $O(M_1^{-1/(2(1+d/s))})$, corresponding to the optimal regression rate for smoothness order s. On the other hand, by choosing K larger one can make the first term as small as desired. Note that the cost of executing the procedure is $O(KM_1^3)$. Then given a computational budget $\mathcal{B}$, one may optimize K and M_1 to get the best possible performance. Clearly, it suffices to choose $K = \log(\mathcal{B})$, hence given the budget $\mathcal{B}$ the performance will be $\tilde{O}(\mathcal{B}^{-1/(6(1+d/s))})$.

6 Illustration

In this section, we use a simple illustrative problem that we call the "sinus world" to investigate the behavior of our regularized fitted Q-iteration algorithm. The problem is designed such that it is especially easy to separate the effect of various sources of difficulties in the learning problem. The state spaces of the "sinus world" is $\mathcal{X} = [-5, 5]$ with the dynamics described in Table 1. An RL agent seeks to maximize its expected discounted sum of rewards in this world. From any state, the agent may take a 0.2-long step to either left or right with some noise added. We used a discount factor $\gamma = 0.8$. The reward function takes the form of a sine function. Below we shall explain how this specific choice allows us to control the difficulty of the problem.

Our regularized fitted Q-iteration algorithm uses the regularized fitting procedure of Eq. (2), and the RKHS defined by the kernel $\mathrm{k}\big((x,a),(x',a')\big) =$

Initial State: $x_0 = -5$

$$\text{Transitions:}\quad x_{t+1} = \begin{cases} \hat{x}_{t+1} & \hat{x}_{t+1} \in (-5, +5), \\ -5 & \hat{x}_{t+1} \le -5, \\ +5 & \hat{x}_{t+1} \ge +5. \end{cases}$$

$$\hat{x}_{t+1} = x_t + a_t + \eta_t \;\; ; \;\; \eta_t \sim \mathcal{N}(0, \sigma_\eta^2)$$

$$; \;\; a_t \in \{-0.2, 0.2\}$$

$$\text{Rewards:}\qquad r(x_t) = \sin(\omega x_t) + \xi_t,\;\; \xi_t \sim \mathcal{N}(0, \sigma_r^2)$$

Table 1. Dynamics of the sinus world. The default parameters are $\omega = 4$, $\sigma_\eta = 0.05$ and $\sigma_r = 1$.

$\mathrm{k}(x, x')\, \mathbb{I}_{\{a=a'\}}$. The state kernel is Gaussian $\mathrm{k}(x, x') = \exp\left(-\|x - x'\|^2 / (2\sigma_{\mathrm{k}}^2)\right)$, and $\mathbb{I}_{\{E\}}$ denotes the indicator function: $\mathbb{I}_{\{L\}} = 1$ if and only if L is true and $\mathbb{I}_{\{L\}} = 0$, otherwise. We used $\sigma_k^2 = 0.1$. In order to gain some speed and numerical stability we decided to use sparsification when solving (2). We used the sparsification method of [5] to selectively add a state-action pair to a set of *dictionary* state-action pairs, which are used as a basis for approximating the full solution. [8] The base distribution used to sample the states X_i is uniform. In all cases we used $K = 50$ iterations and the full dataset in all iterations $(N_1 = \ldots = N_K = 1, M_1 = \ldots = M_k = N)$.

Our aim here is to study the interplay between regularization and the problem parameters such as the frequency of the reward function and the noise of the reward. We will see that both frequency and noise change the difficulty of our RL problem.

In order to understand the effect of reward frequency, note that the difficulty of learning a target function in an RKHS depends on its norm in the same space. With our Gaussian kernel k for a function $f : \mathcal{X} \to \mathbb{R}$ we have $\|f\|_{\mathcal{H}}^2 \propto \int |\hat{f}(\omega)|^2 e^{\sigma_{\mathrm{k}}^2 \omega} d\omega$, where $\hat{f}$ is the Fourier transform of f. Thus, having high frequencies in the target function make the problem difficult. Our target functions have the form TQ, where Q is the element of the RKHS. Operator T can be thought of as smoothing (convolution) in a spatial space which is equivalent to multiplication with the Fourier transform of the probability transition kernel in the Fourier domain. Hence, the RKHS norm of TQ is largely controlled by the energy distribution in the power spectrum of the reward function: When

[8] Sparsification limits the complexity of the model by reducing the degrees of the freedom of the function space, which consequently acts as an implicit regularization and reduces overfitting. This is apparent from the stability of the curves in the following figures as $\lambda \to 0$.

the reward frequency is increased, the problem becomes more difficult. A different source of difficulty is the noise in the reward function that increases sample variance. If we do not use regularization and the signal-to-noise ratio is low, we will fit to the noise instead of the signal.

In these experiments, we evaluate the performance using the relative error $\max_{a\in\mathcal{A}}\left(\frac{\|Q^*(x,a)-Q_K(x,a)\|_2}{\|Q^*(x,a)\|_\infty}\right)$, where the L^2 norm is measured on 1000 points of a regular grid of the state space. The optimal action value function Q^* is calculated using value iteration over the discrete state space resulting from the same regular grid of 1000 points, and Q_K is the estimated action value function after K iterations of our algorithm.

Fig. 2(a) and Fig. 2(b) show the performance of our algorithm as a function of the regularization coefficient λ, for three values of ω and five values of σ_r, respectively. All curves are averaged over 30 repetitions of the experiments. In Fig. 2(a) the results were obtained using $N = 2000$ samples, whereas in Fig. 2(b, the results were obtained using $N = 1000$ samples. On both figures the error bounds are standard error (standard deviation divided by the square-root of the number of runs; here 30).

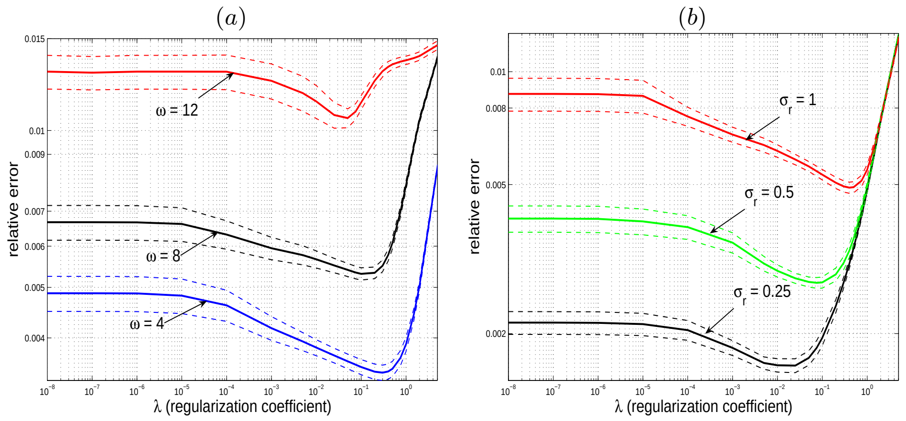

Fig. 2. (a) Effect of changing the reward frequency on the performance of our algorithm. (b) Effect of adding noise to the reward function on the performance of our algorithm.

The results of Fig. 2(a) indicate that increasing the reward frequency increases the generalization error. They also show that for each reward frequency there exists a regularization coefficient λ that attains the minimum error, which is quite pronounced. With an appropriate choice of λ (which can be found by e.g. using a hold-out set) significant savings in computation time are possible (one needs less samples to achieve better results). The same conclusion holds for the case when the reward function is noisy as shown on Fig. 2(b).

In order to gain insight into how the algorithm works we plotted the optimal action-value function for the left action and some action-value functions found by our algorithm for three different values of λ. The result is shown in Fig. 3. In this experiment, we used $N = 200$ samples. We see that for too small values of λ ($\lambda = 10^{-6}$) the procedure overfits, for too large values ($\lambda = 0.5$) it underfits, while for intermediate values ($\lambda = 0.01$) the fit is acceptable.

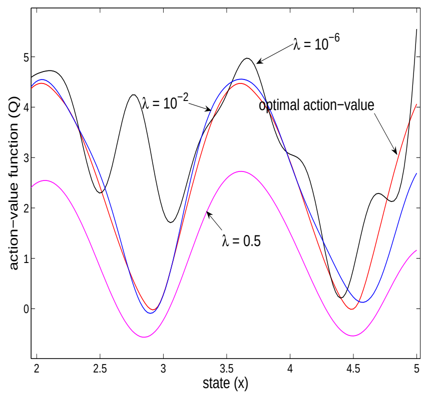

Fig. 3. The optimal action-value function (for the left action) and action-value functions estimated by our algorithm for three values of λ and $N = 200$.

7 Discussion

In this paper we proposed to use penalized least-squares as the regression algorithm in fitted Q-iteration for solving planning problems when a generative model of the environment is available. The main idea is that penalized least-squares is a powerful method of regression, which if used with an appropriate model selection method like cross-validation, can adapt to the difficulty of the regression problem. In this paper we took a step in showing that this is also possible in planning.

In our future work, we plan to investigate fitted Q-iteration in multi-kernel situations (different kernel functions correspond to different smoothness classes). Adapting to the situation when the data lies on a low dimensional sub-manifold of the observation space or when certain variables are irrelevant calls for techniques that allow parameterized kernel families, where ideas from [21] could be useful. A related idea is to use an L^1-penalty in a LASSO-like procedure as an effective feature selection and parameter estimation mechanism (e.g., [4]). Another important research topic is optimally sampling the world. We may use the estimated action-value function in the middle of the fitted Q-iteration procedure to actively choose the most informative samples for the next iteration. Moreover, observing the behavior of this algorithm applied to real-world problems is desirable and we plan to work on it. This request more attention on computational aspects of our method.

Finally, let us note that even though the results of this paper are presented for planning, the extension to the learning scenario when a good policy is to be learned given a long, representative trajectory of some behavior policy seems quite possible along the lines of [2, 1].

References

1. A. Antos, R. Munos, and Cs. Szepesvári, *Fitted Q-iteration in continuous action-space MDPs*, Advances in Neural Information Processing Systems 20 (NIPS-2007), 2008, (in print).
2. A. Antos, Cs. Szepesvári, and R. Munos, *Value-iteration based fitted policy iteration: learning with a single trajectory*, IEEE ADPRL, 2007, pp. 330–337.
3. D. P. Bertsekas and S.E. Shreve, *Stochastic optimal control (the discrete time case)*, Academic Press, New York, 1978.
4. F. Bunea, A. Tsybakov, and M. Wegkamp, *Sparsity oracle inequalities for the lasso*, Electronic Journal of Statistics **1** (2007), 169–194.
5. Y. Engel, S. Mannor, and R. Meir, *The kernel recursive least squares algorithm*, IEEE Transaction on Signal Processing **52** (2004), no. 8, 2275–2285.
6. ———, *Reinforcement learning with Gaussian processes*, ICML '05: Proceedings of the 22nd international conference on Machine learning (New York, NY, USA), ACM, 2005, pp. 201–208.
7. D. Ernst, P. Geurts, and L. Wehenkel, *Tree-based batch mode reinforcement learning*, Journal of Machine Learning Research **6** (2005), 503–556.
8. A. M. Farahmand, M. Ghavamzadeh, Cs. Szepesvári, and S. Mannor, *Regularized policy iteration*, Accepted at the Twenty-Second Annual Conference on Neural Information Processing Systems (NIPS-2008), 2008.
9. L. Györfi, M. Kohler, A. Krzyżak, and H. Walk, *A distribution-free theory of non-parametric regression*, Springer-Verlag, New York, 2002.
10. T. Jung and D. Polani, *Least squares SVM for least squares TD learning*, ECAI, 2006, pp. 499–503.
11. M. Kearns, Y. Mansour, and A.Y. Ng, *A sparse sampling algorithm for near-optimal planning in large Markovian decision processes*, Proceedings of IJCAI'99, 1999, pp. 1324–1331.
12. M.G. Lagoudakis and R. Parr, *Reinforcement learning as classification: Leveraging modern classifiers*, ICML-03, 2003, pp. 424–431.
13. F. A. Longstaff and E. S. Shwartz, *Valuing American options by simulation: A simple least-squares approach*, Rev. Financial Studies **14** (2001), no. 1, 113–147.
14. M. Loth, M. Davy, and P. Preux, *Sparse temporal difference learning using LASSO*, IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning, 2007.
15. S. Mannor, I. Menache, and N. Shimkin, *Basis function adaptation in temporal difference reinforcement learning*, Annals of Operations Research **134** (2005), 215–238.
16. R. Munos and Cs. Szepesvári, *Finite-time bounds for fitted value iteration*, Journal of Machine Learning Research **9** (2008), 815–857.

17. A.Y. Ng and M. Jordan, *PEGASUS: A policy search method for large MDPs and POMDPs*, Proceedings of the 16th Conference in Uncertainty in Artificial Intelligence, 2000, pp. 406–415.
18. D. Ormoneit and S. Sen, *Kernel-based reinforcement learning*, Machine Learning **49** (2002), 161–178.
19. R. Parr, C. Painter-Wakefield, L. Li, and M.L. Littman, *Analyzing feature generation for value-function approximation*, ICML, 2007, pp. 737–744.
20. B. Schölkopf and A.J. Smola, *Learning with kernels*, MIT Press, Cambridge, MA, 2002.
21. N. Srebro and S. Ben-David, *Learning bounds for support vector machines with learned kernels*, COLT, 2006, pp. 169–183.
22. J.N. Tsitsiklis and B. Van Roy, *Optimal stopping of Markov processes: Hilbert space theory, approximation algorithms, and an application to pricing financial derivatives*, IEEE Transactions on Automatic Control **44** (1999), 1840–1851.
23. D-X. Zhou, *Capacity of reproducing kernel spaces in learning theory*, IEEE Transactions on Information Theory **49** (2003), 1743–1752.