Regularized Fitted Q-iteration: Application to 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 planning in a Markovian decision problem, i.e., the problem of finding a good policy given access to a generative model of the environment. We propose to use fitted Q-iteration with penalized (or regularized) least-squares regression as the regression subroutine to address the problem of controlling model-complexity. 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 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

We consider planning in a discounted Markovian Decision Problem (MDP) with continuous state space and finite action space. We assume that transitions can be generated at any selected state for any given action. The algorithm that we consider is fitted Q-iteration (e.g., [8]), an instance of sample-based approximate dynamic programming.

The algorithm's main distinguishing characteristic is that the value function iterates are obtained by solving appropriately defined *regularized* least-squares regression problems. We give the particular form of the algorithm when the value functions considered in the iterations belong to some Reproducing Kernel Hilbert Space (RKHS). Our main theoretical results bound the quality of the solutions as a function of the number of samples used by the algorithm, the relation of the RKHS and the MDP and the number of samples used. As usual when regularization is employed, performance is tuned through the choice of a single scalar parameter, the penalty factor, which, in turn, can be selected in a data dependent manner to optimize the performance.

The rationale of studying the use of regularization in solving MDPs is that regularization has proven to be an extremely effective tool in machine learning, in particular in supervised learning. The main idea underlying regularization is

to achieve model selection by considering the complexity of solution candidates individually. This is done by adding an appropriate complexity penalty, multiplied by the so-called regularization coefficient, to the empirical risk functional. When the reguralization coefficient is chosen in an appropriate way (based on the data or by complexity regularization), *automatic adaptation* to the complexity of the target function becomes possible: The rate of convergence of such a method is almost as fast as if the complexity of the target function was known beforehand (e.g., Theorem 21.2 of [10]).

As the regularization coefficient effectively controls the size of the function space where the solutions are sought in, our approach of using reguralization in fitted Q-iteration can be considered as a way of tuning the function approximator in an approximate dynamic programming procedure. Recently this tuning problem has received considerable attention (e.g., [18, 13, 15, 8, 7, 19]). However, none of the previous works that we know of explored in a systematic manner how regularization influences the performance of the resulting procedure. The only works that we know of that used regularization are that of Jung and Polani [11], Loth et al. [14] and Xu et al. [22]. In particular, Jung and Polani [11] explored penalizing the empirical L^2-norm of the Bellman-residual for finding the value function of a policy given a trajectory in a *deterministic* system, while L^1-penalties for the same problem were considered by Loth et al. [14]. As the straightforward implementation of penalized least-squares involves a nontrivial computational cost, both papers focused on computational efficiency. Xu et al. [22], on the other hand, used sparsification in Least Squares Temporal Difference learning (LSTD) as an implicit form of regularization and studied the performance of the resulting algorithm experimentally. In our more recent work, we analyzed Regularized Policy Iteration methods that use LSTD and a modified version of Bellman Residual Minimization (BRM)and provided finite time performance bounds [9].

Works where planning in generative models were considered include those of Kearns et al. [12] and Ng and Jordan [17]. Our work is complementary: Our method is guaranteed to achieve optimality in the limit (unlike [17] where policy search with a fixed policy class is considered) and it does not scale exponentially with the effective planning horizon (unlike the lookahead tree building method of [12]). However, our method comes with other restrictions: The MDP has to be sufficiently regular in a sense that will be discussed later. The immediate precursor of this work is that of Munos and Szepesvári [16], where fitted value-iteration was studied in the same framework. In contrast to the approach followed here, Munos and Szepesvári considered state-value functions and they did not study regularization. Although our toolkit is the same, due to these differences our results are different than those in [16], as will be further discussed below.

1.1 The organization of the paper

We present the notations and the necessary background on MDPs in Section 2. The fitted Q-iteration algorithm is recalled in Section 3. The first main result that relates the performance of the eventual policy and the L^p-norms of the

errors committed during the iterations is presented in Section 4. This result is in turn used in Section 5 to arrive at specific bounds for L^2 regularization. The behavior of the algorithm and the tradeoffs involved are illustrated on a simple domain 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 of the notation and concepts used in connection to MDPs. We refer the reader to Bertsekas and Shreve [4] for further details in connection to these.

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 probability 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:

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

For brevity, we shall write $\|f\|_\nu$ to denote the $L^2(\nu)$-norm of f. The supremum-norm, $\|f\|_\infty$, of f is defined by $\|f\|_\infty = \sup_{x \in \mathcal{X}} |f(x)|$. 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* with $P(\cdot|x, a)$ being 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 $\mathcal{X}$ is a compact subset of the d-dimensional Euclidean space. We assume that the random immediate rewards are between $-\hat{R}_{\max}$ and $\hat{R}_{\max}$, 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* is specified by a measurable mapping $\pi : \mathcal{X} \to \mathcal{M}(\mathcal{A})$. Such a policy and a random initial state $X_0 \in \mathcal{X}$ gives rise to a random trajectory $(X_t, A_t, R_t)_{t \in \mathbb{N}}$ that we call a *trajectory* of π: Here $A_t \sim \pi(\cdot|X_t)$, $R_t \sim S(\cdot|X_t, A_t)$ and $X_{t+1} \sim P(\cdot|X_t, A_t)$. A policy is deterministic if $\pi(\cdot|x)$ concentrates on a single action for all states $x \in \mathcal{X}$. Such a policy will be identified with a mapping $\pi : \mathcal{X} \to \mathcal{A}$ in the obvious way. 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 incurred while the policy is executed:

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

Here (X_t, A_t, R_t) is a random trajectory underlying π (signified by the use of π as the subindex of the expectation operator in the definition of V^π), where X_0 is such that the support of its distribution is the full state space $\mathcal{X}$ (otherwise this distribution can be chosen arbitrarily). Function V^π is also called the *state-value function* of policy π. Closely related to V^π is the *action-value function* of π:

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

Here the distribution of X_0 is restricted as previously, A_0 is such that at time zero all actions are selected with positive probability everywhere ($\mathbb{P}(A_0 = a | X_0 = x) > 0, (x,a) \in \mathcal{X} \times \mathcal{A}$), $R_0 \sim S(\cdot|X_0, A_0)$, $X_1 \sim P(\cdot|X_0, A_0)$ and otherwise $(X_t, A_t, R_t)_{t \geq 1}$ is a trajectory underlying π. It is easy to see that for any 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,

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

simultaneously for all states $x \in \mathcal{X}$. A policy achieving this goal (i.e., $V^\pi = V^*$) is called an *optimal policy*. Function V^* is called the *optimal value function*.

In order to characterize optimal policies let us define the *optimal action-value function*,

$$Q^*(x,a) = \sup_\pi Q^\pi(x,a), \quad (x,a) \in \mathcal{X} \times \mathcal{A},$$

and the concept of *greedy* policies: A deterministic policy π is *greedy* w.r.t. an action-value function $Q \in B(\mathcal{X} \times \mathcal{A})$ if, for all $x \in \mathcal{X}$ and $a \in \mathcal{A}$, $\pi(x) \in \operatorname{argmax}_{a \in \mathcal{A}} Q(x,a)$. Although greedy policies are non-unique, we will write (by slightly abusing the notation) $\pi = \hat{\pi}(.;Q)$. Because $\mathcal{A}$ is finite, a greedy policy always exists no matter how Q is chosen. The importance of Q^* is that any greedy policy w.r.t. Q^* is optimal. Hence, to find an optimal policy it suffices to determine Q^*.

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

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

As it is well known, T is a contraction operator w.r.t. the supremum-norm with index γ: $\|TQ - TQ'\|_\infty \leq \gamma \|Q - Q'\|_\infty$, $Q, Q' \in B(\mathcal{X})$. Moreover, the optimal action-value function is the unique fixed point of T: $TQ^* = Q^*$. Starting from any $Q_0 \in B(\mathcal{X} \times \mathcal{A})$, $Q_{k+1} = TQ_k$ is thus guaranteed to converge (at an exponential rate) to Q^*. This procedure is called *value iteration*.

Throughout the paper we will use $\mathcal{F} \subset \{ f : \mathcal{X} \to \mathbb{R} \}$ to 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 $f \equiv (f_1, \ldots, f_M)$, $f(x, a_j) = f_j(x)$, $j = 1, \ldots, M$ (note that M denotes the number of actions). The set $\mathcal{F}^M$ will be 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 (e.g., [8]), whose pseudo-code is shown in Fig. 1. The algorithm attempts

```
FittedQ(D,K,Q_0)
// D: samples
// K: number of iterations
// Q_0: Initial action-value function
Q ← Q_0 // Initialization
for k = 0 to K − 1 do
   Q' ← FitQ(Q, D, k)
   Q ← Q'
end for
return Q
```

Fig. 1. Fitted Q-Iteration

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: $D = \{(X_1, A_1, R_1, X'_1), \ldots, (X_N, A_N, R_N, X'_N)\}$. Here, R_t, X'_t are the reward and the next-state when action A_t is chosen in state X_t: $R_t \sim S(\cdot|X_t, A_t)$, $X'_t \sim P(\cdot|X_t, A_t)$. For the sake of simplicity, we assume that the actions are generated by some fixed stochastic stationary policy π_b: $A_t \sim \pi_b(\cdot|X_t)$ and $\{X_t\}$ is an i.i.d. sequence. We will denote the common distribution underlying (X_t, A_t) by ν. 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 the whole state-action space. In particular for this we must have that $\pi_{b0} \stackrel{\text{def}}{=} \min_{a\in\mathcal{A}} \inf_{x\in\mathcal{X}} \pi_b(a|x) > 0$.

The fitting procedure FitQ studied in this paper is *penalized 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 $Q(X_i, A_i)$

[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.

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, which, for any fixed Q_k, in our case is just TQ_k:

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

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 the expressiveness of the function class and its complexity (or the resulting 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 the acceptable complexity of solutions: Choosing larger λ means searching in a smaller space of functions.

When $\mathcal{F}$ is a Sobolev-space[4] of appropriate smoothness order and $\mathrm{Pen}(Q)$ is the corresponding Sobolev-space norm (the L^2-norm of the generalized partials of Q), this optimization leads to thin-plate spline estimates, popular in the non-parametric statistics literature [10]. 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. In order to tune these unknown "parameters" in regression one tries different smoothness orders (this corresponds to choosing the penalty term) with different regularization coefficients and choose the one giving the best empirical error on a hold-out set. The same procedure (though is quite expensive) can be used with fitted Q-iteration. This leads to estimates whose order of rate of convergence is essentially optimal. Further, the convergence rate will scale with the actual roughness, $\mathrm{Pen}(TQ_k)$, of the target function.

Optimizing over a Sobolev-space is a particular case of optimization in a reproducing kernel Hilbert space (RKHS). The latter can be accomplished in a computationally feasible way if one has access to the Mercer kernel function k underlying the RKHS $\mathcal{H}$ and sets $\mathrm{Pen}(Q)$ to be the norm of Q in $\mathcal{H}$ [20]. This way we obtain

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

According to the Representer Theorem (see, e.g., Theorem 4.2 in [20]), any solution to Eq. (2) is the sum of kernels centered on the observed samples: i.e.,

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

[4] Sobolev-spaces generalize Hölder spaces, which in turn put constraints on the pointwise smoothness of functions. In particular, Sobolev-spaces allow functions which are only almost everywhere differentiable. Thus, they can be useful for control problems where value-functions often have ridges.

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{M_k})^\top$ are the coefficients 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 in the expansion of Q_k 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

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

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

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

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

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

The computational complexity of iteration k with a straightforward implementation is $O(M_k^3)$ as it involves the inversion of a matrix. When data is reused between the iterates ($N_k = 1, M_k = N$) only one matrix inversion is necessary as K_k becomes independent of k. However, the total cost as compared when $M_k = N/K$ and when we do K iteration is K^2-times more. Using fast approximate inversion techniques one may get the best of both worlds: Cheaper execution and better use of the samples. In any ways, what remains is to understand how the number of samples influences the quality of the solutions. This is what we study in the next two sections.

4 Error propagation

In order to analyze how the imperfect fitting procedure influences the final error it is customary to rewrite Fitted Q-iteration 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: $\varepsilon_k : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$, $\varepsilon_k = TQ_k - Q_{k+1}$. The "initial error function", ε_{-1}, is introduced for the sake of simplifying some expressions that will follow.

The question studied in this section is 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, provided that we can show that if the functions ε_k are "small" in a sense to be specified below then the final error is also small. For $k \geq 0$ let π_k be a greedy policy w.r.t. Q_k: $\pi_k = \hat{\pi}(\cdot; Q_k)$. With this notation our goal is to bound the norm of $V^* - V^{\pi_K} \geq 0$. In order to arrive at such a bound we need the definition of discounted-average concentrability. The motivation for the definition is that we need to relate the norm of errors under ν (the distribution underlying the samples) to the norm ρ chosen by the user (which could be e.g. the uniform distribution).

Definition 1 (Discounted-average Concentrability of Future-State Distributions) *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

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

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

$$C_{\rho,\nu}^{(2,k)} = (1-\gamma)^2 \sum_{m=0}^{\infty} m \gamma^{m-1} c_{\rho,\nu}(m+k).$$

In general $c_{\rho,\nu}(m)$ diverges to infinity as $m \to \infty$. However, if the rate of divergence of $c_{\rho,\nu}(m)$ is sub-exponential, i.e., if $\Gamma = \limsup_{m \to \infty} 1/m \log c_{\rho,\nu}(m) \leq 0$ then $C_{\rho,\nu}^{(i,j)}$ will be finite. Note that the definition given here is not identical to the previous similar definition by Munos and Szepesvári [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 in [16] state-value functions were considered. Note that it is possible to avoid changing this definition (as it was done in Antos et al. 2), but the price is that the bounds will be more conservative. Interestingly, the bounds here avoid the supremum norm arguments used by Antos et al. [2] and are thus less conservative. Note, however, that the arguments presented here do not extend to continuous action spaces studied in [2].

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,*

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

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 Zhou [23], the following generalization of Theorem 21.1 of Györfi et al. [10] 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 $TQ_k \in \mathcal{H}(= \mathcal{H}_\mathrm{k})$.[5] 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 $L > 0$.[6] Let Q_{k+1} be the solution of (2) with some $\lambda > 0$. Then*

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

holds with probability (w.p.) 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. This choice will depend on the number of samples M_k, the complexity of the target function TQ_k measured by $\|TQ_k\|_{\mathcal{H}}^2$, the dimension d of $\mathcal{X}$, 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 give rise to a better rate – an intuitive result. To find the best λ in a data-dependent manner, one may set up a grid of λs, for any given λ generate a new independent sample and choose the λ that gives the lowest risk estimated on the new sample. If the same λ is used in all iterations then a good value can be selected by again setting up a grid and for any value of λ estimate the performance of the obtained policy by following it from a set of start states generated from ρ and then pick the best policy.

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.

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

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

Corollary 3 (L^2-bound) *Assume that the conditions of the previous theorem hold and that we use the same regularization parameter and 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, $B = \max_{0\le k\le K} \left\|T^k Q_0\right\|_{\mathcal{H}}^2$. Then, for any $\delta > 0$,*

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

holds w.p. at least $1-\delta$ 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 $s/2$. On the other hand, by choosing larger K, one can make the first term as small as desired.

6 Illustration

In this section, we use a simple illustrative problem that we call the "sinus world" to investigate the behavior of the proposed algorithm. The "sinus world" is designed so as to make it is easy to illustrate the role of the choice of the RKHS, the regularization coefficient, the relation of it to the wiggliness of the reward function or how the noise in the dynamics or that in the observed rewards influences the difficulty of the problem. The state space is $\mathcal{X} = [-5, 5]$ and the problem is to navigate an agent to where rewards are large. The agent can move left or right, its actions are noisy and the boundaries of the state space are absorbing. The discount factor is $\gamma = 0.8$. The reward function is a sine function with some frequency ω. For the details see Table 1.

Initial State:	$x_0 = -5$
Transitions:	$x_{t+1} = \begin{cases} \hat{x}_{t+1} & \hat{x}_{t+1} \in (-5, +5), \\ -5 & \hat{x}_{t+1} \le -5, \\ +5 & \hat{x}_{t+1} \ge +5. \end{cases}$
	$\hat{x}_{t+1} = x_t + a_t + \eta_t \;\; ; \;\; \eta_t \sim \mathcal{N}(0, \sigma_\eta^2), \; ; \;\; a_t \in \{-0.2, 0.2\}$
Rewards:	$r(x_t) = \sin(\omega x_t) + \xi_t, \;\; \xi_t \sim \mathcal{N}(0, \sigma_r^2)$

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

We used the regularized fitting procedure of Eq. (2) and the kernel function $\mathrm{k}\big((x,a),(x',a')\big) = \mathrm{k}(x,x')\,\mathbb{I}_{\{a=a'\}}$, where the state kernel is Gaussian $\mathrm{k}(x,x') = \exp\big(-\|x-x'\|^2/(2\sigma_{\mathrm{k}}^2)\big)$ with $\sigma_k^2 = 0.1$.[7]

In order to understand what makes the problem difficult for our procedure remember that we use an RKHS norm as the penalty in our fitting procedure. Thus, we expect performance to deteriorate for problems where the target functions, TQ_k, have a larger RKHS norm. With our choice of the kernel, 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. We see that energies at higher frequencies get exponentially boosted, making the norm to prefer functions with low high frequency content. Now, our target functions have the form $TQ = r + \gamma PMQ$, where $Q \in \mathcal{H}$, $M : B(\mathcal{X}\times\mathcal{A}) \to B(\mathcal{X})$ is an operator defined by $(MQ)(x) = \max_{a\in\mathcal{A}} Q(x,a)$, and P is the Markov kernel underlying the dynamics. Operator M generally increases the high frequency content of its input, but operator P (due to the noise of the dynamics) reduces it. Thus, a noisier dynamics helps to reduce $\|TQ\|_{\mathcal{H}}$. However, a noisier dynamics increases the sample variance (i.e., decreases the signal-to-noise ratio), hence we cannot conclude that a noisier dynamics generally helps. On the other hand, the role of the reward function r is largely clear: $\|TQ\|_{\mathcal{H}}$ can be expected to scale with $\|r\|_{\mathcal{H}}$. For our problem, $\hat{r}(\omega)$ is a Dirac function centered at ω. Thus, a larger ω should give rise to more difficult problems. Another source of difficulty is the noise in the observed rewards. This noise does not decrease $\|TQ\|_{\mathcal{H}}$, but only decreases the signal-to-noise ratio.

When solving (2), for the sake of computational efficiency and to increase numerical stability, we used sparsification. In particular, we used the method of Engel et al. [6] which selectively adds state-action pairs to a set of *dictionary* state-action pairs, which are in turn 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 ($N_1 = \ldots = N_K = 1, M_1 = \ldots = M_k = N$).

Performance is evaluated by the relative error, $\max_{a\in\mathcal{A}} \big(\frac{\|Q^*(x,a) - Q_K(x,a)\|_2}{\|Q^*(x,a)\|_\infty}\big)$. Here the L^2 norm is estimated on a grid of 1000 points evenly spaced in the state space. An approximation to the optimal action-value function Q^* is calculated by discretizing the problem using the same regular grid.

Fig. 2(a) (Fig. 2(b)) show the performance of our algorithm as a function of the regularization coefficient λ, for three values of ω (resp., σ_r). All curves are averaged over 30 independent runs of the experiments (across algorithms the same random seeds were used). 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).

[7] $\mathbb{I}_{\{E\}}$ denotes the indicator function: $\mathbb{I}_{\{L\}} = 1$ if and only if L is true and $\mathbb{I}_{\{L\}} = 0$, otherwise.

[8] Sparsification limits the complexity of the fitted functions and consequently acts as implicit regularization, thus reducing the chance of overfitting. This is also apparent from the stability of the curves in the following figures as $\lambda \to 0$.

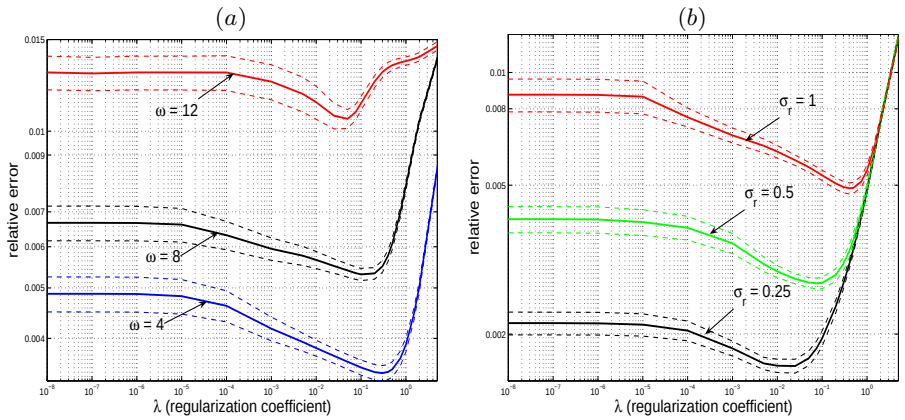

Fig. 2. (a) Effect of changing the reward frequency on the performance of our algorithm. (b) Effect of adding noise to the reward function on the performance of our algorithm.

The results of Fig. 2(a) confirm 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. The minimum is pronounced. Thus, with an appropriate choice of λ (which can be found by e.g. using a hold-out set) significant saving in computation time is possible as one needs less samples to achieve better results. The same conclusion holds for the case when the noise of the observed rewards is varied (Fig. 2(b)).

In order to gain further insight into the behavior of the algorithm we plotted the optimal action-value function for the left action along with the action-value functions found by our algorithm for three different values of λ (Fig. 3). In this experiment, we used $N = 200$ samples. We see that when λ is too small ($\lambda = 10^{-6}$) the procedure overfits, while if λ is too large ($\lambda = 0.5$), underfitting happens. Finally, an intermediate value ($\lambda = 0.01$) gives acceptable fits.

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 problem addressed is that in fitted value iteration and other sample-based planning methods for small sample sizes the function space has to be chosen not only to fit the MDP but also to control over- or underfitting.

As one main contribution of the paper we analyzed the finite-sample performance of the proposed procedure. Although finite-sample performance of fitted value iteration has been considered earlier [1, 2], 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 it. The analysis presented here builds on these previous works, but extends and improves them.

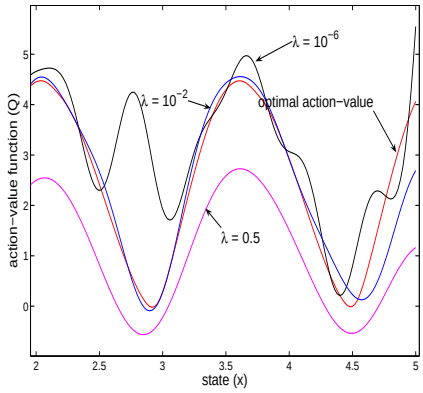

Fig. 3. The optimal action-value function (for the left action) and action-value functions estimated by our algorithm for three values of λ and $N = 200$.

As 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. For such a situation ideas of Srebro and Ben-David [21] could be useful. Feature selection could also be addressed by introducing an L^1-penalty in a LASSO-like procedure (e.g., [5]). Another important research topic is to optimize the sample distribution. One idea is to use the estimated action-value function while running the algorithm to actively choose the most informative samples for the next iteration. It would also be desirable to gain experience by applying the proposed method in some realistic problems. Currently, one main limitation is that the procedure is quite expensive to run.

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 looks possible along the lines of the works [1, 2, 3].

Bibliography

[1] A. Antos, Cs. Szepesvári, and R. Munos. Value-iteration based fitted policy iteration: learning with a single trajectory. In *IEEE ADPRL*, pages 330–337, 2007.

[2] A. Antos, R. Munos, and Cs. Szepesvári. Fitted Q-iteration in continuous action-space MDPs. In *Advances in Neural Information Processing Systems 20 (NIPS-2007)*, 2008. (in print).

[3] A. Antos, Cs. Szepesvári, and R. Munos. Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning*, 71:89–129, 2008.

[4] D. P. Bertsekas and S.E. Shreve. *Stochastic Optimal Control (The Discrete Time Case)*. Academic Press, New York, 1978.

[5] F. Bunea, A. Tsybakov, and M. Wegkamp. Sparsity oracle inequalities for the lasso. *Electronic Journal of Statistics*, 1:169–194, 2007.

[6] Y. Engel, S. Mannor, and R. Meir. The kernel recursive least squares algorithm. *IEEE Transaction on Signal Processing*, 52(8):2275–2285, 2004.

[7] Y. Engel, S. Mannor, and R. Meir. Reinforcement learning with Gaussian processes. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 201–208, New York, NY, USA, 2005. ACM. ISBN 1-59593-180-5. doi: http://doi.acm.org/10.1145/1102351.1102377.

[8] D. Ernst, P. Geurts, and L. Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

[9] A. M. Farahmand, M. Ghavamzadeh, Cs. Szepesvári, and S. Mannor. Regularized policy iteration. In *Advances in Neural Information Processing Systems 21 (NIPS-2008)*, 2008. (to appear).

[10] L. Györfi, M. Kohler, A. Krzyżak, and H. Walk. *A distribution-free theory of nonparametric regression*. Springer-Verlag, New York, 2002.

[11] T. Jung and D. Polani. Least squares SVM for least squares TD learning. In *ECAI*, pages 499–503, 2006.

[12] M. Kearns, Y. Mansour, and A.Y. Ng. A sparse sampling algorithm for near-optimal planning in large Markovian decision processes. In *Proceedings of IJCAI'99*, pages 1324–1331, 1999.

[13] M.G. Lagoudakis and R. Parr. Reinforcement learning as classification: Leveraging modern classifiers. In *ICML-03*, pages 424–431, 2003.

[14] M. Loth, M. Davy, and P. Preux. Sparse temporal difference learning using LASSO. In *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: 215–238, 2005.

[16] R. Munos and Cs. Szepesvári. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9:815–857, 2008.

[17] A.Y. Ng and M. Jordan. PEGASUS: A policy search method for large MDPs and POMDPs. In *Proceedings of the 16th Conference in Uncertainty in Artificial Intelligence*, pages 406–415, 2000.

[18] D. Ormoneit and S. Sen. Kernel-based reinforcement learning. *Machine Learning*, 49:161–178, 2002.

[19] R. Parr, C. Painter-Wakefield, L. Li, and M.L. Littman. Analyzing feature generation for value-function approximation. In *ICML*, pages 737–744, 2007.

[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. In *COLT*, pages 169–183, 2006.

[22] X. Xu, D. Hu, and X. Lu. Kernel-based least squares policy iteration for reinforcement learning. *IEEE Trans. on Neural Networks*, 18:973–992, 2007.

[23] D-X. Zhou. Capacity of reproducing kernel spaces in learning theory. *IEEE Transactions on Information Theory*, 49:1743–1752, 2003.