Regularized Fitted Q-iteration for Planning in Continuous-Space Markovian Decision Problems

Amir massoud Farahmand Mohammad Ghavamzadeh Csaba Szepesvári Shie Mannor

***Abstract*— Reinforcement learning with linear and non-linear function approximation has been studied extensively in the last decade. However, as opposed to other fields of machine learning such as supervised learning, the effect of finite sample has not been thoroughly addressed within the reinforcement learning framework. In this paper we propose to use L^2 regularization to control the complexity of the value function in reinforcement learning and planning problems. We consider the Regularized Fitted Q-Iteration algorithm and provide generalization bounds that account for small sample sizes. Finally, a realistic visual-servoing problem is used to illustrate the benefits of using the regularization procedure.**

I. 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 and a complexity penalty, the 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. When the parameter is chosen in an appropriate way (for example by cross-validation or 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. [10]).

Recently the problem of tuning function approximators has received considerable attention for the solution of Markov Decision Processes, especially in the reinforcement learning (RL) community. For example, [14] considered parameterized function approximation architecture where the parameters are changed to better minimize the Bellman residual error, and [18] constructs new basis functions from the Bellman residual in fitted value iteration. In other approaches, non-parametric regression is used where the function representation has the potential to adapt to the actual difficulty of the problem. Examples of this approach include [12] where support vector machines are used to represent policies in an approximate policy iteration procedure, the tree-regression based fitted Q-iteration algorithm of [8], or the GPTD algorithm of [7] that builds on Gaussian processes regression.

In this work, we consider a non-parametric regression approach based on penalized least-squares regression method. Even though penalized least-squares regression is one of the most successful approaches to supervised regression, it is surprising that it has not been thoroughly investigated in RL. Work similar to ours includes [11] and [13]. They do not, however, provide an explicit performance analysis like this work. Moreover, [11] just works for the deterministic transitions with a fixed policy, whereas we consider the control problem with stochastic transitions.

Here we extend the fitted Q-iteration algorithm of [8] and let it use penalized least-squares, a regularization-based algorithm, for fitting value functions. This way we borrow the strength of a state-of-the-art supervised learning approach to help solve learning and planning problems more efficiently. We call this algorithm Regularized Fitted Q-Iteration (RFQI). We develop specific formulae for kernel-based RFQI. Our main theoretical results bound the quality of the solutions found 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 regularization coefficient. We argue that non-trivial performance gains are possible if one chooses the regularization coefficient in a data-dependent manner. We show this empirically in our experiments. Although finite-sample performance of fitted Q-iteration has been considered earlier [1], to our best knowledge this work alongside [9] are the first work that address finite-sample performance of a *regularized* RL algorithm.

Both planning and learning can benefit from regularization. In many real world planning problems of interest, the simulation time is limited and a policy has to be found relatively quickly. In such problems, a simulator is available to generate samples from a typically high dimensional state space. For example, in control of complex networks (e.g., power and communication networks), the only way to compute a good policy is through simulation. Simulating a complex network is computationally demanding since it requires a discrete events simulations [16]. Using RFQI algorithm that takes the finiteness of the available data into account is therefore relevant for planning as well as learning.

II. BACKGROUND AND NOTATION

We briefly review a few concepts and notations from analysis and Markovian Decision Processes (MDP). 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 defined as

This work was partially supported by NSERC and AIF.

A.m. Farahmand, M. Ghavamzadeh and Cs. Szepesvári are with the Dept. of Computing Science, University of Alberta, Edmonton, AB T6G 2E8, Canada. Cs.Sz. is on leave from MTA SZTAKI, Bp., Hungary. {amir,mgh,szepesva}@cs.ualberta.ca

S. Mannor is with the Dept. of Electrical & Computer Eng., McGill University, Montreal, QC H3A 2A7, Canada shie.mannor@mcgill.ca

$\|f\|_{p,\nu}^p = \int |f(s)|^p \nu(ds)$. 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 $\mathcal{X}$ is a compact subset of the d-dimensional Euclidean space. We assume the expected immediate rewards $r(x, a) = \int r S(dr|x, a)$ are bounded by $R_{\max}$: $\|r\|_\infty \le R_{\max}$.

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, i.e.

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

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, $A_t \sim \pi(\cdot|X_t)$, while if π is deterministic then 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

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

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. The policy that attains the optimal value function *any* state $x \in \mathcal{X}$ is called the optimal policy, i.e. if $V^\pi(x) = V^*(x)$ for all $x \in \mathcal{X}$. Note that the optimal policy is not necessarily unique. In order to characterize optimal policies, it will be useful to define the optimal action-value function, $Q^*(x, a)$:

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

Fig. 1. Fitted Q-Iteration

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

Further, we say that a deterministic policy π is *greedy* w.r.t. (with respect to) 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 $a \in \mathcal{A}$, $\pi(x) \in \operatorname{argmax}_{a \in \mathcal{A}} Q(x, a)$. The Bellman optimality operator $T : B(\mathcal{X} \times \mathcal{A}) \to B(\mathcal{X} \times \mathcal{A})$ is defined by

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

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: $TQ^* = Q^*$. 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 $\mathcal{F} \subset \{ f : \mathcal{X} \to \mathbb{R} \}$ 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.

III. 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. Because computing the effect of the Bellman operator applied to an action-value function 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:

$$D = \{(X_1, A_1, R_1, X_1'), \ldots, (X_N, A_N, R_N, X_N')\}.$$

In this paper 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: $A_t \sim \pi_b(\cdot|X_t)$, $X_t' \sim P(\cdot|X_t, A_t)$, $R_t \sim S(\cdot|X_t, A_t)$. 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 $\pi_{b0} \overset{\text{def}}{=} \min_{a \in \mathcal{A}} \inf_{x \in \mathcal{X}} \pi_b(a|x) > 0$.

The fitting procedure that we study in this paper is penalized 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

$$Q_{k+1} = \operatorname*{argmin}_{Q \in \mathcal{F}^M} \left[\hat{L}_k(Q) + \lambda \mathrm{Pen}(Q) \right], \qquad (1)$$

where

$$\hat{L}_k(Q) = \frac{1}{M_k} \sum_{i=N_k}^{N_k+M_k-1} \left[R_i + \gamma \max_{a' \in \mathcal{A}} Q_k(X'_i, a') - Q(X_i, A_i) \right]^2,$$

and $\mathrm{Pen}(Q)$ is a penalty term and $\lambda > 0$ is the regularization coefficient.[1] The first term is the sample-based least-squares error of using Q 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

$$L_k(Q) = \mathbb{E}\left[\left(R_i + \gamma \max_{a' \in \mathcal{A}} Q_k(X'_i, a') - Q(X_i, A_i) \right)^2 \right].$$

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 infinity the empirical loss converges to L_k and therefore, we hope that the iterate Q_{k+1} converges to TQ_k. For assuring that, one needs to prevent overfitting or over-smoothing. This is the job of the second term on the right hand side of (1). This term regulates the complexity of solutions. Choosing a larger λ means searching in a smaller space of functions and vice versa.

Considering the discussion in the previous paragraph, a viable approach is choosing a large (possibly infinite dimensional) space $\mathcal{F}^M$ and using regularization to control its complexity. When $\mathcal{F}^M$ is a Sobolev-space 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 non-parametric statistics literature [10]. Nevertheless, Sobolev space is not the only possibility. It is a particular case of a reproducing kernel Hilbert space (RKHS). In an RKHS, we 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 [19]. This way we obtain

$$Q_{k+1} = \operatorname*{argmin}_{Q \in \mathcal{H}} \left[\hat{L}_k(Q) + \lambda \left\| Q \right\|_{\mathcal{H}}^2 \right]. \qquad (2)$$

According to the Representer Theorem (e.g., see [19]), 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

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

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

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

where $Z_j = (X_j, A_j)$, $Z^{(k)}_j = (X'_j, A^{(k)}_j)$,

$$A^{(k)}_j = \operatorname*{argmax}_{a \in \mathcal{A}} Q_k(X'_j, a),$$

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.

The choice of the regularization coefficient and the RKHS itself is still a problem. For the case of Sobolev space, this corresponds to the choice of the smoothness order of the space. We need a model selection procedure to select these parameters. The approach common in regression can be followed here, too: Try different smoothness orders (or the parameter that describes the RKHS) 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, $\mathrm{Pen}(TQ_k)$.

IV. Error Propagation

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

$$Q_{k+1} = TQ_k - \varepsilon_k \quad (k \ge 0); \quad \text{and} \quad \varepsilon_{-1} = Q^* - Q_0. \qquad (4)$$

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 $\{Q_k\}_k$ and not vice versa (except for ε_{-1}, the "initial error", which is introduced just for notational simplification). 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

[1] Note that in practice one would generate samples as-needed basis, 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.

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 $V^* - V^{\pi_K}$ (because of the definition of the optimal value function, this quantity is guaranteed to be non-negative).

Recall that ν denotes the distribution underlying $\{(X_t, A_t)\}$. For the sake of flexibility, we allow the user to choose another distribution, $\rho \in \mathcal{M}(\mathcal{X})$, that is used in assessing the procedure's performance, e.g. the stationary distribution induced by the optimal policy. The main result of this section is the following theorem that bounds the loss of using the learned policy, measured by $\|\cdot\|_{p,\rho}$, as a function of the losses of the solutions of the regression problems solved while running the algorithm, measured by $\|\cdot\|_{p,\nu}$. The proof is omitted from this paper.

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 there exist constants constants $C_{\rho,\nu}^{(1,1)}$ and $C_{\rho,\nu}^{(2,1)}$ that only depend on ρ, ν, γ and the MDP dynamics such that

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

Theorem 1 shows that if the error sequence ε_k is small, the error between the optimal value function V^* and the value of our estimated policy V^{π_K} is small too. We can use standard PAC results to give a high probability bound on the magnitude of each ε_k (and consequently $\overline{\varepsilon}_K$) as a function of the available data. We provide such a bound for the case of RKHS in the next section.

Theorem 1 suggests a model selection mechanism: Since our goal is to minimize $\overline{\varepsilon}_K$, we can use different regularization coefficients and/or kernel parameter in solving Eq. (1) in such a way that all ε_k remains as small as possible. Although we cannot calculate ε_k directly, we can still use the empirical norm as an estimation of its norm. This can be done by doing a cross-validatoin at each iteration. Model selection at each iteration is important because the appropriate regularization coefficient and even the function space (which is determined by the kernel parameter) may change during iterations.

V. L^2-BOUND FOR REGURALIZED KERNEL-BASED REGRESSION

In this section we assume that Q_{k+1} is obtained by solving the RKHS regularization problem of Eq. (2). The following result can be obtained by generalizing Theorem 21.1 of [10] to arbitrary RKHS with smooth kernel functions, combining it with Prop. 3 of [20]. 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. In the following theorem, we assume that $X_t \sim \nu_{\mathcal{X}}$ is an i.i.d. sequence and $A_t \sim \pi_b(\cdot|X_t)$ for some π_b that selects all actions with non-zero probability. This assumption basically means that we have access to the generative model, and is the case of *planning*. However, this assumption is not essential, and we just use it to simplify the proof. We can extend this result to the case that the agent observes a single trajectory generated by a fixed policy by having appropriate mixing condition on the MDP, i.e. learning case (see [1]).

Theorem 2: Assume that $\mathcal{X} = [0,1]^d$, $\mathrm{k} \in \mathrm{Lip}^*(s, C(\mathcal{X}, \mathcal{X}))$, $s > d$, and Q_k is such that $TQ_k \in \mathcal{H}(= \mathcal{H}_{\mathrm{k}})$.[2] 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 $L > 0$.[3] Let Q_{k+1} be the solution of (2) with some $\lambda > 0$. Then

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

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 TQ_k measured by $\|TQ_k\|_{\mathcal{H}}^2$, the dimension of the problem d, and some notion of smoothness measured by s. With $\lambda = cM_k^{-1/(1+d/s)}$ the rate of convergence is $O(M_k^{-1/(1+d/s)})$. In fact, when we have a Sobolev space with smoothness degree κ, one can show $s = 2\kappa$ and this rate will be the optimal rate for regression for smoothness κ. 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 $B = \max_{0 \leq k \leq K} \left\| T^k Q_0 \right\|_{\mathcal{H}}^2$. Then, for any $\delta > 0$ with probability at least $1 - \delta$,

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

where $C = (1-\gamma)(C_{\rho,\nu}^{(1,1)})^{\frac{1}{2}} + \gamma (C_{\rho,\nu}^{(2,1)})^{\frac{1}{2}}$ and $c_1, c_2, c_3 > 0$ are universal constants.

Again, 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 $s = 2\kappa$. On the other hand, by letting K approach infinity, 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 $\tilde{O}(\mathcal{B}^{-1/(6(1+d/s))})$.

[2] For the definition of the generalized Lipschitz space Lip^* see [20].

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

VI. VISUAL-SERVOING PROBLEM

By considering a visual-servoing problem as our experiment, we study the effect of regularization coefficient and kernel parameter on the performance of the RFQI algorithm. Also we compare its performance with a conventional visual-servoing controller.

Visual servoing is the task of controlling the motion of a robot using vision data ([5]). Visual data can include inputs like the position of the end-effector on visual input or the image of an external object on the robot-mounted camera. Forward kinematic model of the robot, cameras' parameters, and the relative position of objects in the world define a visual-motor kinematic model $X = f(q)$ where $q \in \mathbb{R}^d$ is joint variables for a robot with d degrees of freedom and $X \in \mathbb{R}^m$ is the vector of visual features. The aim of visual servoing is to find a control signal $u(t)$ that changes q over time so that some objective function is minimized (e.g. $e(t) = X(t) - X^*$) goes to zero asymptotically.

A conventional controller design methodology uses local model of visual-motor kinematic to design the controller. Defining the visual-motor Jacobian as $J(q) = \frac{\partial f(q)}{\partial q}$, the control signal would be $u(t) = \dot{q}(t) = -J^\dagger(q)$ where $J^\dagger(q)$ is the pseudo-inverse of the Jacobian at $q(t)$. Nevertheless, there are at least two problems with conventional controllers. One is that they need to know the dynamics of the system. There are adaptive methods that can partially remedy this problem. The other more important problem is that they are usually local controllers and cannot benefit from long horizon plans. Therefore, one can expect that their performance would not be optimal because of their myopic design.

In these experiments we apply the RFQI method to design controller for the visual-servoing task. The problem is visual set-point regulation for the *Puma 560* robotic arm with an eye-to-hand stationary stereo rig configuration. The visual features are the coordinate of end-effector onto the image space of these two cameras ($X \in \mathbb{R}^4$) and the set-point is $X^* = [0\ 0\ 0\ 0]^T$. We have access to 3 degrees of freedom of the robot and the discrete-time control signal is $u(t) \in \{-1, +1\}^3$ with time step of 0.02sec. We desire to minimize the number of steps to the goal from arbitrary initial position. We formulate this problem as solving a discounted MDP with $\gamma = 0.95$ where

$$R_t = \begin{cases} -1, & \text{if } ||X_{t+1} - X^*||^2 > 10, \\ \frac{1}{1+||X_{t+1}-X^*||^2}, & \text{otherwise.} \end{cases}$$

This encourages the robot to move towards the goal as soon as possible.

For all experiments, we use i.i.d. samples from $q \in U((-1,1)^3)$ for both training and policy evaluation. Actions in the training set are generated uniformly. In our experiments, we re-use the same data in all iterations. The iterations are limited to $K = 1000$, but if the action-value function Q_{k+1} is very close to Q_k (empirical norm smaller than 10^{-5}), we stop the iteration. We use a Gaussian kernel $\mathrm{k}((q_1, u_1), (q_2, u_2)) = \exp(-\frac{||q_1 - q_2||^2}{2\sigma^2})\mathbb{I}_{\{u_1 = u_2\}}$ with different kernel parameters σ^2 in all experiments. We use MatLab, Corke's Robotics Toolbox [6], and the Epipolar Geometry Toolbox [15].

The goal of the first experiment is studying the effect of the kernel parameter σ and the regularization coefficient λ on the performance of the FQI. We evaluate the performance of $\hat{\pi}(\cdot; Q_K)$, the greedy policy w.r.t. Q_K, by a Monte Carlo method using 5000 random trajectories starting from a initial positions chosen uniformly at random and then following the policy.

Fig. 2 shows the performance of policies generated by RFQI as a function of the regularization coefficient and the kernel parameter. Lighter regions show better performance and darker regions show worse performance. A prominent region where the performance is considerably better than other regions is evident. This region has moderate values of λ and σ^2. The performance degradation for very small values of λ is an indication of over-fitting. Under-fitting is also observable for large values of regularization coefficient. Although the performance is less sensitive to the kernel parameter than to the regularization coefficient, poor performance is visible for very small values of σ^2. Such values of σ^2 lead to over-fitting.

Fig. 3 compares the performance of RFQI equipped with a empirical Bernstein race model selection mechanism (see [17]) to the performance of a conventional controller when the number of training samples is varying.

The problem formulation for the RL problem is a variation of the total time to the goal. To compare these two controllers, we need to make sure both controllers use the same amount of power. The RL agent's policy selects the control signal $u(t) \in \{-1, +1\}^3$. Therefore, the power of the signal is 3. For the conventional controller (which is a linear controller), we normalize its output so that its power become 3. When the error is large, this modification prevents the controller to spend so much power. When the error is very small, it makes the controller act like a switching controller.

For this experiment, the number of samples is changing from around 700 to 7000. We apply RFQI for 20 models (different λs and σs), and use the model selection to choose one of them. For model selection, we put a maximum limit of 1000 trajectory samples for each model (refer to [17] for details). After selecting the best model, we compare it with the conventional controller by evaluating the return of 1000 randomly selected trajectory paths. We run this experiment 9 times. The error bars (or dotted interval) shows the standard error around the empirical average.

The result shown in Fig. 3 indicates that the RFQI with a model selection procedure generates competitive policies. Even when the number of samples is not so large, it performs better than the conventional controller. Note that we do not necessarily claim that our controller is *better* than the usual practice in visual-servoing research since we have not tried hard to optimize the conventional controller, but the claim is that the RFQI's solution performs comparably well without using any domain specific knowledge.

In summary, we observed that RFQI that just uses samples can perform quite well without explicitly using the dynamics

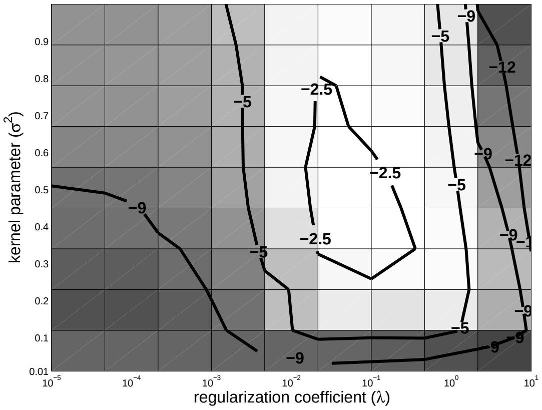

Fig. 2. The average return of policies computed using RFQI as a function of the regularization coefficient (λ) and the kernel parameter (σ^2).

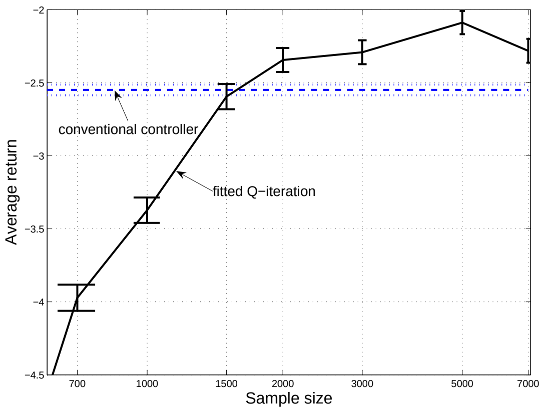

Fig. 3. Comparison of RFQI's policy with a conventional controller.

of system. This is important in visual-servoing since in many cases we do not have access to visual-motor kinematic model, but we have access to the robot and can get samples from it. Also we noticed that the performance of the resulted policy not only depends on the efficient use of data (which a method like RFQI with its optimal convergence rate can do well), but also depends on selecting the right function space. A regularization-based method like RFQI lets us partially solve this problem by reducing a part of model selection problem to the selection of the regularization coefficient. This emphasizes the importance of effective model selection.

VII. Discussion

In this paper we proposed to use penalized least-squares as the regression algorithm used in fitted Q-iteration for solving learning and planning problems. The main idea is that penalized least-squares is a powerful method of regression which, when used with a model selection mechanism, can adapt to the difficulty of the regression problem. By applying this method to a visual-servoing problem, we showed that it can give competitive results for a real-world problem.

In our future work, we intend to extend the proof to the case of learning. Also efficient model selection mechanism for the case of single trajectory path needs more attention. Adapting to the situation when the data lies on a low dimensional sub-manifold of the observation space or when certain variables are irrelevant is an important issue for many real-world problems. Using L^1-penalty in a LASSO-like procedure (e.g., [4]) may prove to be useful. 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. Finally, let us note that the extension of our results to the learning scenario when the data consists of a representative trajectory of some behavior policy looks possible along the lines of [2].

References

[1] 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).

[2] 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.

[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*, 1:169–194, 2007.

[5] F. Chaumette and S. Hutchinson. Visual servo control, part I: Basic approaches. *IEEE Robotics and Automation Magazine*, 13(4):82–90, December 2006.

[6] P.I. Corke. A robotics toolbox for MATLAB. *IEEE Robotics and Automation Magazine*, 3(1):24–32, March 1996.

[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.

[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 Sh. Mannor. Regularized policy iteration. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 441–448. 2009.

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

[13] 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.

[14] S. Mannor, I. Menache, and N. Shimkin. Basis function adaptation in temporal difference reinforcement learning. *Annals of Operations Research*, 134:215–238, 2005.

[15] G.L. Mariottini and D. Prattichizzo. Egt: a toolbox for multiple view geometry and visual servoing. *IEEE Robotics and Automation Magazine*, 3(12), December 2005.

[16] S. P. Meyn. *Control Techniques for Complex Networks*. Cambridge, 2008.

[17] V. Mnih, Cs. Szepesvári, and J.-Y. Audibert. Empirical Bernstein stopping. In *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)*, pages 672–679, 2008.

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

[19] B. Schölkopf and A.J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

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