# Robot Learning with Regularized Reinforcement Learning

**Amir-massoud Farahmand**[1], **Mohammad Ghavamzadeh**[2], **Csaba Szepesvári**[1], **Shie Mannor**[3]
[1]Department of Computing Science, University of Alberta, Canada {amir,szepesva}@cs.ualberta.ca

[2]INRIA Lille - Nord Europe, Team SequeL, France mohammad.ghavamzadeh@inria.fr

[3]Department of ECE, McGill University, Canada - Department of EE, Technion, Israel shie.mannor@mcgill.ca

## 1. Introduction

Sequential decision-making problems are of prominent importance in robotics. Many robotics tasks, such as motion planning for a robotic arm, gait optimization for a quadruple robot, and dynamic balancing of humanoids, can all be formulated as a sequential decision-making problem. To give a concrete example, consider the visual-servoing problem where the goal is finding a control signal that moves the robot's arm from the initial position to the target position while minimizing constraints such as consumed energy, movement time, and the jerkiness of the movement. In visual-servoing, both current and target positions are directly or indirectly specified by visual features. The problem becomes more difficult in uncalibrated setting where the kinematic and dynamic models of the robot-camera system are not known a priori. This may happen, for instance, when the robot picks up a wooden stick with unknown measurements and wants to use it to move another objects around – a common scenario for humans and some other animals.

Reinforcement learning (RL) and dynamic programming (DP) are mathematical frameworks that cast a sequential decision-making task as an optimization problem. A common approach for solving RL/DP and finding optimal or close-to-optimal policies is using an intermediate entity, called *value* function. In many robotics tasks, the state space is large or infinite, and thus, the value function should inevitably be represented by a function approximator, whose quality has a major impact on the quality of the learned policy. Although different methods for value function approximation have been considered in RL/DP literature, such as generalized linear models with predefined basis functions, kernel regression, regression trees, and neural networks, the designer still needs to make nontrivial design choices such as basis function selection or the stopping rule for growing the tree. The hassle of designing the right function approximator is one important reason that has prevented roboticist from using RL/DP methods in their problems. On the other hand, the usual practice of hand-designing controllers is not easily scalable to problems with highly-nonlinear and uncertain dynamics.

Our goal is to design adaptive RL/DP methods that automatically find the right value-function approximator for the problem in hand. The decision must ideally be based on the *regularities* of the problem such as the smoothness or sparsity of the value function, and the number of samples coming from the agent's direct interaction with the environment. We are advocating the use of *regularization* as a powerful tool to design adaptive learning procedures. The use of regularization has been proven useful in supervised learning, and we believe the same is true for RL/DP. The idea of regularization is to start with a large function space and control the solution's complexity by a regularization (penalization) term. This is a principled way to balance approximation and estimation errors (or bias and variance).

In this work, we present $L^2$-regularized counterparts of two widely-used RL/DP algorithms: *approximate policy iteration* and *fitted Q-iteration*. We show how our regularized algorithms can be implemented efficiently when the value function approximator belongs to 1) a space spanned by a finite number of linearly independent basis functions (a parametric approach), and 2) a reproducing kernel Hilbert space (a non-parametric approach). We also prove finite-sample performance bounds for our algorithms. In particular, we show that they are able to achieve rates that are as good as the corresponding regression rates when the value functions belong to a known smoothness class. For the detailed formulation of the algorithms and their implementation and analysis, please refer to Farahmand et al. (2009b) for regularized policy iteration, and to Farahmand et al. (2009a) for regularized fitted Q-itteration. The results of applying our algorithms to a visual-servoing problem can be found in Farahmand et al. (2009c).

## 2. Regularized Policy Iteration

The pseudo-code of the policy iteration algorithm is shown in Algorithm 1. In this algorithm, a policy is first evaluated (Line 5) and then improved (Line 6). This process is repeated for $N$ steps. In each iteration $i$, training sample $\mathcal{D}_i = \{(X_t, A_t, R_t)\}_{1\le t\le n}$ is generated by policy $\pi_i$, thus, $A_t = \pi_i(X_t)$ and $R_t \sim \mathcal{R}(\cdot|X_t, A_t)$, and is given to an approximate policy evaluation (APEval) method. The goal of APEval is to find a "close enough" approximation of the value function of policy $\pi_i$, $Q^{(i)}$. The improved policy $\pi_{i+1}$ is the greedy policy w.r.t. the action-value function estimate $Q^{(i)}$. Two widely-used APEval methods in RL/DP are *Bellman residual minimization* (BRM) and *least-squares temporal-difference learning* (LSTD). In our regularized policy iteration algorithms we replace these two APEval methods with their $L^2$-regularized counterparts, REG-BRM and REG-LSTD.

In REG-BRM, the action-value function of policy $\pi_i$ is estimated by solving the following coupled optimization problems:

$$h^*(\cdot;Q) = \underset{h\in\mathcal{F}^M}{\operatorname{argmin}} \left[ \left\| h - \hat{T}^{\pi_i} Q \right\|_n^2 + \lambda_{h,n} J(h) \right],$$
$$Q^{(i)} = \underset{Q\in\mathcal{F}^M}{\operatorname{argmin}} \left[ \left\| Q - \hat{T}^{\pi_i} Q \right\|_n^2 - \left\| h^*(\cdot;Q) - \hat{T}^{\pi_i} Q \right\|_n^2 + \lambda_{Q,n} J(Q) \right],$$

where $Z_t = (X_t, A_t)$ and $Z'_t = \big(X_{t+1}, \pi_i(X_{t+1})\big)$, $(\hat{T}^{\pi_i}Q)(Z_t) = R_t + \gamma Q(Z'_t)$ is the empirical Bellman operator, $\|\cdot\|_n^2$ is the empirical norm, and $J(\cdot)$ and $\lambda_{h,n}, \lambda_{Q,n} > 0$ are the regularization (penalty) term and coefficients, respectively.

In REG-LSTD, the following coupled optimization problems must be solved to estimate the action-value function of the ith policy:

$$h^*(\cdot;Q) = \underset{h\in\mathcal{F}^M}{\operatorname{argmin}} \left[ \left\| h - \hat{T}^{\pi_i} Q \right\|_n^2 + \lambda_{h,n} J(h) \right],$$
$$Q^{(i)} = \underset{Q\in\mathcal{F}^M}{\operatorname{argmin}} \left[ \left\| Q - h^*(\cdot;Q) \right\|_n^2 + \lambda_{Q,n} J(Q) \right].$$

The choice of $\mathcal{F}^M$ is by the designer. It can be a finite dimensional parametric space with $L^2$ norm of weights as the regularizer, or an infinite dimensional function space like a reproducing kernel Hilbert space (RKHS) with its corresponding norm as the regularizer, i.e. $J(\cdot) = \|\cdot\|_{\mathcal{H}}^2$. In these two cases, we can show that the coupled optimization problems of REG-BRM and REG-LSTD have closed-form solutions.

## 3. Regularized Fitted Q-Iteration

Fitted Q-iteration is an approximate value iteration method, whose pseudo-code is shown in Algorithm 2.

**Algorithm 1** Approximate Policy Iteration

```
1: ApproxPolicyIteration($N$, $Q^{(-1)}$, APEval)
     • $N$: number of iterations
     • $Q^{(-1)}$: Initial action-value function
2: $\pi_0 \leftarrow \hat{\pi}(Q^{(-1)})$
3: for $i = 0$ to $N-1$ do
4:     Generate training sample $\mathcal{D}_i$
5:     $Q^{(i)} \leftarrow$ APEval$(\pi_i, \mathcal{D}_i)$    // an approximate policy
       evaluation method, e.g., REG-LSTD or REG-BRM
6:     $\pi_{i+1} \leftarrow \hat{\pi}(Q^{(i)})$    // the greedy policy w.r.t. $Q^{(i)}$
7: end for
8: return $Q^{(N-1)}$ or $\pi_N = \hat{\pi}(Q^{(N-1)})$
```

We add regularization to fitted Q-iteration by using a regularized least-squares algorithm as its fitting procedure. The algorithm starts with an initial action-value function $Q^{(0)}$. At each iteration $i$, it applies an approximate Bellman optimality operator and uses a regularized least-squares algorithm to fit $Q^{(i+1)}$ as the new estimate of the action-value function. Assuming that $\mathcal{D}_i$ contains $n_i$ samples, the $(i+1)^{\text{th}}$ action-value function is obtained by

$$Q^{(i+1)} = \underset{Q\in\mathcal{F}^M}{\operatorname{argmin}} \frac{1}{n_i} \sum_{j=1}^{n_i} \left[ R_j + \gamma \max_{a'\in\mathcal{A}} Q^{(i)}(X'_j, a') - Q(X_j, A_j) \right]^2 + \lambda J(Q),$$

where $J(Q)$ and $\lambda > 0$ are the regularization term and coefficient, respectively, and $X'_j$ is the next-state of $X_j$. As in REG-BRM and REG-LSTD, we can obtain closed-form solution when $\mathcal{F}^M$ is the space spanned by a finite number of basis functions or an RKHS.

**Algorithm 2** Fitted Q-Iteration

```
1: Fitted Q-Iteration($N$, $Q^{(0)}$, FitQ)
     • $N$: number of iterations
     • $Q^{(0)}$: Initial action-value function
2: for $i = 0$ to $N-1$ do
3:     Generate training sample $\mathcal{D}_i$
4:     $Q^{(i+1)} \leftarrow$ FitQ$(Q^{(i)}, \mathcal{D}_i)$    // a fitting procedure,
       e.g., penalized least-squares
5: end for
6: return $Q^{(N)}$
```

## References

Farahmand, A. M., Ghavamzadeh, M., Szepesvári, C., & Mannor, S. (2009a). Regularized fitted Q-iteration for planning in continuous-space Markovian decision problems. *ACC*.

Farahmand, A. M., Ghavamzadeh, M., Szepesvári, C., & Mannor, S. (2009b). Regularized policy iteration. *NIPS21* (pp. 441–448).

Farahmand, A. M., Shademan, A., Jägersand, M., & Szepesvári, C. (2009c). Model-based and model-free reinforcement learning for visual servoing. *ICRA*.