Regularization in Reinforcement Learning

Amir-massoud Farahmand[1], **Mohammad Ghavamzadeh**[2], **Csaba Szepesvári**[1], **Shie Mannor**[3] *
[1]Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada
[2]INRIA Lille - Nord Europe, Team SequeL, France
[3]Department of ECE, McGill University, Canada - Department of EE, Technion, Israel

1 Introduction

The use of function approximator to represent the value function of Reinforcement Learning (RL) and Dynamic Programming (DP) problems with large state spaces is inevitable. Although different methods for value function approximation have been considered (such as generalized linear model with predefined basis functions, regression trees, neural networks, etc.), the designer still needs to make non-trivial design choices such as basis function selection or the stopping rule for growing the tree. These choices, however, should ideally depend on the characteristics of problem in hand such as the number of available samples, the smoothness, and the sparsity of the true value function. We would like a learning method to automatically **adapt** to the actual difficulty of the problem based on the training data.

Regularization technique is a powerful and flexible approach to design adaptive learning procedures. The idea of regularization is that one starts from a large function space and controls the solution's complexity by a regularization (or penalization) term. This is a principled and flexible way to balance approximation and estimation errors. Here, the notion of complexity depends on the function space and the form of regularizer, and may have smoothness, sparsity, or other well-defined interpretations.

Even though regularization technique has been used in supervised learning, their use in RL/DP is new. We develop regularized counterparts of standard **Approximate Value Iteration** and **Approximate Policy Iteration** algorithms. Our statistical analyses show that these methods have an almost **optimal finite-sample sample-complexity convergence rate** for value function estimation. We briefly review these methods in the following sections. Please refer to [1; 2] for more information.

2 Regularized Policy Iteration

The ability to evaluate a given policy is the core requirement to run policy iteration. We need a policy evaluation procedure that estimates the action-value function of a given policy π as accurate as possible. Regularization helps us develop flexible policy evaluation methods. In this section, we present **REG-LSTD** and **REG-BRM** as regularized extensions of standard Least-Squares Temporal Difference (LSTD) and Bellman Residual Minimization (BRM) algorithms.

The generic algorithm for approximate policy iteration is shown in Table 1. When ApproxPolicyEval$(\cdot)$ function is an implementation of either REG-LSTD or REG-BRM, this algorithm is called *Regularized Policy Iteration*. In the i^{th} iteration, we use training samples $D_i = \{(X_t, A_t, R_t)\}_{1 \le t \le n}$ $(1 \le i \le K)$, generated by a policy π, thus, $A_t = \pi(X_t)$ and $R_t \sim \mathcal{R}(\cdot|X_t, A_t)$. Policy π, the behavior policy, should be exploratory enough. $\pi_i = \hat{\pi}(\cdot; Q^{(i-1)})$ is the greedy policy with respect to the most recent estimate of action-value function $Q^{(i-1)}$.

In REG-BRM, the estimated action-value function at iteration i for policy π_i is computed by solving the following coupled optimization problems:

*Csaba Szepesvári is on leave from MTA SZTAKI. This research was funded in part by the National Science and Engineering Research Council (NSERC), iCore and the Alberta Ingenuity Fund.

```
**ApproxPolicyIteration**($K$,$Q^{(-1)}$,ApproxPolicyEval)
// $K$: number of iterations
// $Q^{(-1)}$: Initial action-value function
// ApproxPolicyEval: Approximate policy evaluation procedure (e.g. REG-LSTD or REG-BRM)
**for** $i = 0$ to $K-1$ **do**
  $\pi_i(\cdot) \leftarrow \hat{\pi}(\cdot; Q^{(i-1)})$    // the greedy policy w.r.t. $Q^{(i-1)}$
  Generate training sample $D_i$
  $Q^{(i)} \leftarrow$ ApproxPolicyEval($\pi_i, D_i$)
**end for**
**return** $Q^{(K-1)}$ or $\pi_K(\cdot) = \hat{\pi}(\cdot; Q^{(K-1)})$
```

Table 1: Approximate Policy Iteration algorithm

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

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

In REG-LSTD, we must solve the following coupled optimization problem:

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

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 the corresponding norm for regularization term, i.e. $J(\cdot) = \|\cdot\|_{\mathcal{H}}^2$. In these cases, the coupled optimization problem has a closed-form solution.

3 Regularized Fitted Q-Iteration

Regularized Fitted Q-Iteration is a regularized instance of the fitted Q-Iteration algorithm, an approximate value iteration method, that attempts to approximate the optimal action-value function iteratively. The algorithm starts from an initial action-value function $Q^{(0)}$. At each iteration k ($k = 0, \cdots$), it applies an approximate Bellman optimality operator and uses a regularized least-squares algorithm to fit $Q^{(k+1)}$ as the new estimate of the action-value function.

Assuming that in the k^{th} iteration we use m_k samples with index $n_k \le i < n_k + m_k = n_{k+1} - 1$, the $(k+1)^{\text{th}}$ iterate is obtained by

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

where $J(Q)$ is the regularization term and $\lambda > 0$ is the regularization coefficient. As in REG-LSTD and REG-BRM, $\mathcal{F}^M$ can be a parametric space or an RKHS. In these cases, we have closed-form solution. Another possibility is working with wavelets basis or an over-complete dictionary and choose $J(\cdot)$ as l_1 norm of the weights. This choice of function space and regularizer can be helpful when the action-value function has spatially-varying smoothness.

References

[1] Amir-massoud Farahmand, Mohammad Ghavamzadeh, Csaba Szepesvári, and Shie Mannor. Regularized fitted q-iteration for planning in continuous-space markovian decision problems. In *Proceedings of American Control Conference (ACC) (Accepted for publication)*, 2009.

[2] Amir-massoud Farahmand, Mohammad Ghavamzadeh, Csaba Szepesvári, and Shie 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.