MS&E 338: Reinforcement Learning
Lectures on Approximate Dynamic Programming

Mohammad Ghavamzadeh*
mohammad.ghavamzadeh@inria.fr

February 9 & 11, 2015

1 Approximate Dynamic Programming (ADP)

Dynamic programming (DP) algorithms are the most powerful tools to solve a reinforcement learning (RL) problem, i.e., to find an optimal policy. However, these algorithms guarantee to find an optimal policy only if the environment (i.e., the dynamics and the rewards) is completely known and the size of the state and action spaces are not too large. When one of these conditions is violated, e.g.,

- the state space $\mathcal{X}$ and/or action space $\mathcal{A}$ are large or infinite,
- the model of the system (the transition probability P and reward r functions) is unknown and the only information about the environment is of the form of samples of transitions and rewards,
- we do not have enough time and/or sample to compute the quantity of interest at each iteration k of the DP algorithm (i.e., $\mathcal{T}V_{k-1}$ for VI and $V^{\pi_{k-1}}$ for PI algorithms),

approximate algorithms are needed, and thus, DP methods turn to approximate dynamic programming (ADP) and RL algorithms. Unfortunately, in this case, the convergence and performance guarantees of the standard DP algorithms are no longer valid, and the main theoretical challenge is to study the performance of ADP and RL algorithms, which is the main focus of these lectures.

Assume that at the k'th iteration of value iteration (VI), we are not able to compute $\mathcal{T}V_{k-1}$, and have its approximation instead. As a result, the next value function V_k won't be $\mathcal{T}V_{k-1}$ as in the standard VI, but will be an approximation of this quantity, i.e., $V_k \approx \mathcal{T}V_{k-1}$. Thus, we can no longer use the max-norm contraction property of the Bellman optimality operator and write

$$||V^* - V_k||_\infty \leq \gamma ||V^* - V_{k-1}||_\infty.$$

This property is at the heart of the proof of the standard VI, and thus without it, it is not possible to prove the convergence of the approximate value iteration (AVI) algorithm to the optimal value function.

Similarly, assume that at the k'th iteration of policy iteration (PI), we are not able to compute the value of the current policy $V^{\pi_{k-1}}$, and have its approximation $\widehat{V}^{\pi_{k-1}} \approx V^{\pi_{k-1}}$ instead. As a result, the next policy

*Many thanks to Ian Osband for his notes on these lectures.

generated by PI, i.e., the greedy policy w.r.t. $\widehat{V}^{\pi_{k-1}}$, is no longer the greedy policy w.r.t. π_{k-1}, i.e.,

$$\pi_k(x) = \arg\max_{a \in \mathcal{A}} \left[r(x,a) + \gamma \sum_{x' \in \mathcal{X}} P(x'|x,a) \widehat{V}^{\pi_{k-1}}(x') \right] \neq (\mathcal{G}\pi_{k-1})(x),$$

and thus, we cannot guarantee that $V^{\pi_k} \geq V^{\pi_{k-1}}$, and maintain the monotonically improving behavior of the standard PI. The monotonically improving property is at the heart of the proof of the standard PI, and thus without it, it is not possible to prove the convergence of the approximate policy iteration (API) algorithm to an optimal policy.

2 Approximate Value Iteration (AVI)

Assume that we can approximate the Bellman optimality operator up to ϵ accuracy, i.e., at iteration k, we have

$$V_{k+1} = \widehat{\mathcal{T}} V_k \qquad \text{such that} \qquad V_{k+1} = \mathcal{T} V_k + \epsilon_k \quad \text{or} \quad ||V_{k+1} - \mathcal{T} V_k||_\infty = \epsilon_k.$$

Is there any hope in this AVI scheme: $V_{k+1} = \widehat{\mathcal{T}} V_k$? By hope we mean whether the bounded error at each iteration ϵ_k, remains bounded when it propagates through the iterations of the AVI algorithm (after we run AVI for K iterations. The following theorem shows that there is hope and in fact this simple idea is a valid ADP algorithm.

Theorem 1. *We run the above AVI scheme for K iteration and return V_K. If π_K is the greedy policy w.r.t. V_K, i.e., $\pi_K = \mathcal{G} V_K$, then we have*

$$\|V^* - V^{\pi_K}\|_\infty \leq \frac{2\gamma}{(1-\gamma)^2} \underbrace{\max_k \epsilon_k}_{\text{error at each iteration}} + \frac{2\gamma^{K+1}}{1-\gamma} \underbrace{\|V^* - V_0\|_\infty}_{\text{error of initialization}}.^1$$

Proof. We first bound $\|V^* - V_{k+1}\|_\infty$ as

$$\|V^* - V_{k+1}\|_\infty \leq \|\mathcal{T} V^* - \mathcal{T} V_k\|_\infty + \|\mathcal{T} V^k - \mathcal{T} V_{k+1}\|_\infty \leq \gamma \|V^* - V_k\|_\infty + \epsilon_k.$$

If we repeat this process recursively, we will obtain

$$\|V^* - V_K\|_\infty \leq \sum_{k=0}^{K-1} \gamma^{K-1-k} \epsilon_k + \gamma^K \|V^* - V_0\|_\infty \leq \frac{1}{1-\gamma} \max_k \epsilon_k + \gamma^K \|V^* - V_0\|_\infty.$$

Now we need to connect $\|V^* - V_K\|_\infty$ to $\|V^* - V^{\pi_K}\|_\infty$. Since $\pi_K = \mathcal{G} V_K$, we may write

$$\mathcal{T}^{\pi_K} V_K = \mathcal{T} V_K \geq \mathcal{T}^{\pi^*} V_K,$$

and thus, we may write

$$\begin{aligned} V^* - V^{\pi_K} &= \mathcal{T}^{\pi^*} V^* - \mathcal{T}^{\pi^*} V_K + \underbrace{\mathcal{T}^{\pi^*} V_K}_{\leq \mathcal{T}^{\pi_K} V_K} - \mathcal{T}^{\pi_K} V^* + \mathcal{T}^{\pi_K} V^* - \mathcal{T}^{\pi_K} V^{\pi_K} \\ &\leq \mathcal{T}^{\pi^*}(V^* - V_K) + \mathcal{T}^{\pi_K}(V_K - V^*) + \mathcal{T}^{\pi_K}(V^* - V^{\pi_K}). \end{aligned}$$

So, we have

$$\|V^* - V^{\pi_K}\|_\infty \leq 2\gamma \|V^* - V_K\|_\infty + \gamma \|V^* - V^{\pi_K}\|_\infty \leq \frac{2\gamma}{1-\gamma} \|V^* - V_K\|_\infty$$

This together with the earlier bound on $\|V^* - V_k\|_\infty$ completes the result. □

[1] You see the dependence of the bound on γ, the closer it is to 1, the looser the bound is.

So, the error ϵ_k does not blow up when it propagates through the iterations of the algorithm, and thus, it makes sense to design an ADP algorithm based on this simple idea.

Reference: For more information on the propagation of error and the final performance in AVI, see

- R. Munos. Performance bounds in ℓ_p-norm for approximate value iteration. SIAM Journal of Control and Optimization, 46(2):541-561, 2007.

2.1 An AVI Algorithm / Fitted Q-Iteration (or Fitted Value Iteration in general)

Here is the algorithmic implementation of this simple idea

$$Q_{k+1} = \widehat{\mathcal{T}} Q_k.$$

Algorithm

At each iteration k,

- Given a data set $\mathcal{D}_k = \{x_i, a_i, r_i = r(x_i, a_i), x_i'\}_{i=1}^N$ such that

  $$(x_i, a_i) \sim \rho \quad \text{and} \quad x_i' \sim P(\cdot|x_i, a_i),$$

  where ρ is a distribution over $\mathcal{X} \times \mathcal{A}$.
- Using $\mathcal{D}_k$, we build a new data set of $\mathcal{D}_k' = \{(x_i, a_i), y_i\}_{i=1}^N$, where

  $$y_i = r_i + \gamma \max_{a'} Q_k(x_i', a_i') = (\widehat{\mathcal{T}} Q_k)(x_i, a_i).$$

- $Q_{k+1} = \textbf{Fit}(f, \mathcal{F}, \mathcal{D}_k')$.

What is happening here?

We have a

$$\begin{aligned} \text{Target Function} &= \mathcal{T} Q_k \qquad \text{and} \\ \text{Noisy Observation} &= \widehat{\mathcal{T}} Q_k \ \text{ at a finite number of } N \text{ points} \end{aligned}$$

We minimize the *empirical error*

$$Q_{k+1} = \widehat{Q} = \min_{f \in \mathcal{F}} \|f - \widehat{\mathcal{T}} Q_k\|_{\widehat{\rho}}$$

in this **fitting** process ($\widehat{\rho}$ is the empirical measure of ρ – N samples from ρ), with the goal of minimizing the *true error*

$$Q = \min_{f \in \mathcal{F}} \|f - \mathcal{T} Q_k\|_{\rho}.$$

This is the *regression problem* of $Q_{k+1} \approx \mathcal{T} Q_k$. and thus, our objective is

$$\|\widehat{Q} - \mathcal{T} Q_k\|_\rho \leq \underbrace{\|\widehat{Q} - Q\|_\rho}_{\text{estimation error}} + \underbrace{\|Q - \mathcal{T} Q_k\|_\rho}_{\text{approximation error}}$$

to be small.

Controlling these two sources of error is similar to the classic bias/variance tradeoff that you see in supervised learning. Making one smaller makes the other larger, so we should find a good tradeoff.

Main Message

A supervising learning algorithm, namely regression, is sitting at the heart or at the inner-loop of this AVI algorithm. So, any regression algorithm can be plugged in for this fitting procedure such as linear regression, neural network, random forest, etc.

Although a supervising learning algorithm is at the heart of AVI, AVI and in general the problem of control and RL is a more difficult problem than supervised learning because

- Due to its iterative nature, AVI contains several supervised learning (regression) problems, one per iteration.
- Due to its iterative nature, it is important to see that how much the error at each iteration grows when it propagates through the iterations of the algorithm.
- Choice of the sampling distribution ρ plays an important role in the final performance of the algorithm. This is equivalent to the important problem of *exploration* in the batch setting.

Reference: For more information on the Fitted Q-iteration, see

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

Reference: For more information on finite-sample analysis of Fitted value-iteration, see

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

3 Approximate Policy Iteration

Each iteration k of PI contains the following two tasks:

1. **Policy Evaluation** (PE): $V^{\pi_k} = \mathcal{T}^{\pi_k} V^{\pi_k}$
2. **Policy Improvement** (PI): $\pi_{k+1} = \mathcal{G} V^{\pi_k} = \mathcal{G}\pi_k$ (the second equality is by abuse of notation)

where we define the bellman operator of policy π_k as

$$\mathcal{T}^{\pi_k} f(x) = r\big(x, \pi_k(x)\big) + \gamma \sum_{x'} P\big(x'|x, \pi_k(x)\big) f(x'),$$

and the greedification operator as

$$(\mathcal{G}f)(x) = \arg\max_a \left[r(x,a) + \gamma \sum_{x'} P(x'|x,a) f(x') \right].$$

We may have the following approaches to develop an approximate policy iteration (API) algorithm:

(i) Minimizing the **Policy Evaluation Error**:

$$V_k = V^{\pi_k} + \epsilon_k \qquad \text{or} \qquad \|V_k - V^{\pi_k}\|_\infty = \epsilon_k.$$

(ii) Minimizing the **Bellman Residual Error**:

$$V_k = \mathcal{T}^{\pi_k} V_k + \epsilon_k \qquad \text{or} \qquad \|\mathcal{T}^{\pi_k} V_k - V_k\|_\infty = \epsilon_k.$$

(iii) **Classification-based PI**: the classification approach to PI

$$\pi_{k+1} = \mathcal{G}\pi_k + \epsilon_k,$$

which is formally defined as

$$\ell_{\pi_k}(\pi_{k+1}) = \epsilon_k = \mathcal{T}V^{\pi_k} - \mathcal{T}^{\pi_{k+1}} V^{\pi_k} \qquad \text{or} \qquad \|\mathcal{T}V^{\pi_k} - \mathcal{T}^{\pi_{k+1}} V^{\pi_k}\|_\infty = \epsilon_k.$$

3.1 Minimizing the Policy Evaluation Error

The first question is: Is there any hope for this procedure?

Theorem 2.

$$\limsup_{k\to\infty} \|V^* - V^{\pi_k}\|_\infty \leq \frac{2\gamma}{(1-\gamma)^2} \limsup_{k\to\infty} \overbrace{\|V^{\pi_k} - V_k\|_\infty}^{\epsilon_k}.$$

Proof. ***Reference:*** See

- Chapter 6.2 of D. Bertsekas and J. Tsitsiklis. Neuro-Dynamic Programming. Athena Scientific, 1996.

□

This result also holds in ℓ_p-norm w.r.t. appropriate measures.

Reference: For this see

- R. Munos. Error bounds for approximate policy iteration. In Proceedings of the Twentieth International Conference on Machine Learning, pages 560-567, 2003.

Since the *policy evaluation error* at each iteration remains bounded when we iterate, it makes sense to derive an API algorithm based on this simple idea.

Algorithm

At each iteration k of the algorithm,

1. We build the training set $\mathcal{D}_k = \{x_i\}_{i=1}^N, \qquad x_i \sim \rho.$
2. $\forall x_i \in \mathcal{D}_k$ and some $a \in \mathcal{A}$, we do M rollouts to estimate

$$Q^{\pi_k}(x_i, a) \approx \widehat{Q}^{\pi_k}(x_i, a) = \frac{1}{M} \sum_{j=1}^{M} R_j^{\pi_k}(x_i, a),$$

where $R_j^{\pi_k}(x_i, a)$ is the j'th rollout estimate for state-action pair (x_i, a), i.e, the cumulative discounted sum of rewards up to some horizon H (this is also known as *return*) of starting at state x_i, first taking action a and then following policy π_k. The horizon H can be considered ∞ or $\approx \frac{1}{1-\gamma}$. In these cases, $R_j^{\pi_k}(x_i, a)$ is an unbiased estimate of $Q_k^{\pi}(x_i, a)$ (in the second approximately, but very reasonable approximation). If H is smaller than $\frac{1}{1-\gamma}$, then we have to add this rollout error to the rest of the approximation errors of the algorithm.

3. Form $\mathcal{D}_k$, we build the new data set $\mathcal{D}'_k = \left\{\left((x_i, a_i), \widehat{Q}^{\pi_k}(x_i, a_i)\right)\right\}_{i=1}^{\geq N}$.
4. $\widetilde{Q}^{\pi_k} = \textbf{Fit}(f, \mathcal{F}, \mathcal{D}'_k)$

Similar to the fitted Q-iteration algorithm, this is the *regression* problem $\widetilde{Q} \approx Q^{\pi_k}$, in which we have *noisy observations* $\widehat{Q}^{\pi_k}$ at a finite number of points (more than N) of the *target function* Q^{π_k}, and we try to minimize the *empirical error*

$$\widetilde{Q}^{\pi_k} = \min_{f \in \mathcal{F}} \|f - \widehat{Q}^{\pi_k}\|_{\widehat{\rho}},$$

with the hope of minimizing the true error

$$Q = \min_{f \in \mathcal{F}} \|f - Q^{\pi_k}\|_{\rho}.$$

So we have the same break down of error

$$\|Q^{\pi_k} - \widetilde{Q}\|_{\rho} \leq \underbrace{\|Q^{\pi_k} - Q\|_{\rho}}_{\text{approximation error}} + \underbrace{\|Q - \widetilde{Q}\|_{\rho}}_{\text{estimation error}}.$$

We can use any standard regression technique for the fitting part of this algorithm. Note that this is not a very popular and practical API algorithm but it is very useful for theoretical study of these algorithms.

3.2 Classification-based Policy Iteration

The first question is: Is there any hope for this procedure?

Theorem 3. *If π_K is the policy returned by the algorithm after K iterations, then we have*

$$\|V^* - V^{\pi_K}\|_{\infty} \leq \gamma^k \underbrace{\|V^* - V^{\pi_0}\|_{\infty}}_{\text{error of initialization}} + \frac{1-\gamma^K}{(1-\gamma)^2} \underbrace{\max_k \epsilon_k}_{\text{error at each iteration}}.$$

Proof.

$$\begin{aligned} V^* - V^{\pi_{k+1}} &= \mathcal{T}V^* - \mathcal{T}V^{\pi_k} + \mathcal{T}V^{\pi_k} - \mathcal{T}^{\pi_{k+1}}V^{\pi_k} + \mathcal{T}^{\pi_{k+1}}V^{\pi_k} - \mathcal{T}^{\pi_{k+1}}V^{\pi_{k+1}} \\ \Longrightarrow \|V^* - V^{\pi_{k+1}}\|_{\infty} &\leq \gamma\|V^* - V^{\pi_k}\|_{\infty} + \epsilon_k + \gamma\|V^{\pi_k} - V^{\pi_{k+1}}\|_{\infty}. \end{aligned} \tag{1}$$

We now bound $\|V^{\pi_k} - V^{\pi_{k+1}}\|_{\infty}$ as

$$\begin{aligned} V^{\pi_k} - V^{\pi_{k+1}} &= \mathcal{T}^{\pi_k}V^{\pi_k} - \mathcal{T}^{\pi_{k+1}}V^{\pi_k} + \mathcal{T}^{\pi_{k+1}}V^{\pi_k} - \mathcal{T}^{\pi_{k+1}}V^{\pi_{k+1}} \\ \Longrightarrow \|V^{\pi_k} - V^{\pi_{k+1}}\|_{\infty} &\leq \gamma\|V^{\pi_k} - V^{\pi_{k+1}}\|_{\infty} + \epsilon_k \\ \Longrightarrow \|V^{\pi_k} - V^{\pi_{k+1}}\|_{\infty} &\leq \frac{\epsilon_k}{1-\gamma}. \end{aligned} \tag{2}$$

From (1) and (2), we have

$$\|V^* - V^{\pi_{k+1}}\|_{\infty} \leq \gamma\|V^* - V^{\pi_k}\|_{\infty} + \frac{\epsilon_k}{1-\gamma}. \tag{3}$$

The statement of the theorem is proved by recursively applying (3). □

Since the *classification-based error* at each iteration remains bounded as we iterate, it makes sense to derive an API algorithm based on this idea.

Algorithm

At each iteration k of the algorithm,

1. We build a data set $\mathcal{D}_k = \{x_i\}_{i=1}^N, \qquad x_i \sim \rho.$
2. $\forall x_i \in \mathcal{D}_k$ and $\forall a \in \mathcal{A}$, we do M rollouts to estimate

$$Q^{\pi_k}(x_i, a) \approx \widehat{Q}^{\pi_k}(x_i, a) = \frac{1}{M} \sum_{j=1}^{M} R_j^{\pi_k}(x_i, a).$$

3. Form $\mathcal{D}_k$ we build the data set $\mathcal{D}'_k = \big\{\big(x_i, \arg\max_a \widehat{Q}^{\pi_k}(x_i, a)\big)\big\}_{i=1}^N$.
4. $\pi_{k+1} = \textbf{Class}(f, \Pi, \mathcal{D}'_k)$.

What is happening here?

We minimize the *empirical error*

$$\pi_{k+1} = \widehat{\pi} = \min_{f \in \Pi} \widehat{\mathcal{L}}(f, \pi_k, \widehat{\rho}),$$

with the target of minimizing the *true error*

$$\pi = \min_{f \in \Pi} \mathcal{L}(f, \pi_k, \rho).$$

This is the *classification* problem $\pi_{k+1} \approx \mathcal{G}\pi_k$, in which the objective is

$$\mathcal{L}(\widehat{\pi}, \pi_k, \rho) \leq \underbrace{\mathcal{L}(\widehat{\pi}, \pi, \rho)}_{\text{estimation error}} + \underbrace{\mathcal{L}(\pi, \pi_k, \rho)}_{\text{approximation error}},$$

to be small.

The question now is what is the *error function* $\mathcal{L}$. We define $\mathcal{L}$ and its empirical version $\widehat{\mathcal{L}}$ based on the *loss function* ℓ as follows:

$$\mathcal{L}(f, \pi_k, \rho) = \int \ell_{\pi_k}(f, x)\rho(dx) \qquad \text{and} \qquad \widehat{\mathcal{L}}(f, \pi_k, \widehat{\rho}) = \frac{1}{N} \sum_{i=1}^{N} \ell_{\pi_k}(f, x_i), \quad x_i \sim \rho,$$

where the *loss function* ℓ is either

$$\textbf{0/1 loss} \qquad \ell_{\pi_k}(f, x) = \mathbb{1}\big\{f(x) \neq \arg\max_a Q^{\pi_k}(x, a)\big\},$$

or

$$\textbf{weighted loss} \qquad \ell_{\pi_k}(f, x) = \max_a Q^{\pi_k}(x, a) - Q^{\pi_k}\big(x, f(x)\big).$$

It has been theoretically shown that the weighted loss function is more appropriate for deriving this class of API algorithms.

For algorithms of this class, see

Reference:

- M. Lagoudakis and R. Parr. Reinforcement learning as classification: Lever- aging modern classifiers. In Proceedings of the Twentieth International Conference on Machine Learning, pages 424-431, 2003.
- A. Fern, S. Yoon, and R. Givan. Approximate policy iteration with a policy language bias: Solving relational Markov decision processes. Journal of Artificial Intelligence Research, 25:85-118, 2006.
- A. Lazaric, M. Ghavamzadeh, and R. Munos. Analysis of a classification- based policy iteration algorithm. In Proceedings of the Twenty-Seventh International Conference on Machine Learning, pages 607-614, 2010.

For the analysis of the algorithms of this class, see ***Reference:***

- A. Lazaric, M. Ghavamzadeh, and R. Munos. Analysis of a classification- based policy iteration algorithm. In Proceedings of the Twenty-Seventh International Conference on Machine Learning, pages 607-614, 2010.
- V. Gabillon, A. Lazaric, M. Ghavamzadeh, and B. Scherrer. Classification- based policy iteration with a critic. In Proceedings of the Twenty-Eighth International Conference on Machine Learning, pages 1049-1056, 2011.
- B. Scherrer, M. Ghavamzadeh, V. Gabillon, and M. Geist. Approximate modified policy iteration. In Proceedings of the Twenty-Ninth International Conference on Machine Learning, pages 1207-1214, 2012.

For the application of this class of algorithms, see ***Reference:***

- V. Gabillon, M. Ghavamzadeh, and B. Scherrer. Approximate dynamic programming finally performs well in the game of Tetris. In Proceedings of Advances in Neural Information Processing Systems 26, 2013.

3.3 Dealing with Bellman Residual

In AVI and both approaches to API that we have discussed so far, the inner-loop of the corresponding ADP algorithm is a supervised learning problem, either regression or classification. In this section, we look at a relatively different class of ADPs in which we have a *fixed-point problem* at the heart of the algorithm.

The first question is: Is there any hope for this procedure?

Theorem 4.

$$\limsup_{k\to\infty} \|V^* - V^{\pi_k}\|_\infty \leq \frac{2\gamma}{(1-\gamma)^2} \limsup_{k\to\infty} \overbrace{\|\mathcal{T}^{\pi_k} V_k - V_k\|_\infty}^{\epsilon_k}.$$

This result also holds in ℓ_p-norm w.r.t. appropriate measures.

Reference: For this see

- R. Munos. Error bounds for approximate policy iteration. In Proceedings of the Twentieth International Conference on Machine Learning, pages 560-567, 2003.

3.3.1 Bellman Residual Minimization (BRM)

The goal here is to solve the following *empirical* problem

$$\widehat{Q}^{\pi_k} = \min_{f \in \mathcal{F}} \underbrace{\frac{1}{N} \sum_{i=1}^{N} \left[f(x_i, a_i) - \overbrace{\Big(r(x_i, a_i) + \gamma f\big(x_i', \pi_k(x_i')\big) \Big)}^{\widehat{\mathcal{T}}^{\pi_k} f} \right]}_{\widehat{L}_N(f, \pi_k)}$$

with the hope of solving the following problem:

$$Q = \min_{f \in \mathcal{F}} \underbrace{\|f - \mathcal{T}^{\pi_k} f\|_{2,\rho}^2}_{L(f, \pi_k)}.$$

Unfortunately, there is a negative result here that unless we do double sampling, i.e., , we have

$$\mathbb{E}\left[\hat{L}_N(f, \pi_k)\right] \neq L(f, \pi_k).$$

In fact, we have

$$\mathbb{E}\left[\hat{L}_N(f, \pi_k)\right] = \Big(f(x, a) - (\mathcal{T}^{\pi_k} f)(x, a)\Big)^2 + \mathbf{Var}\Big(R + \gamma f\big(X', \pi_k(X')\big)\Big).$$

The proof of this inequality and a potential solution can be found at

Reference:

- 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 Journal, 71:89-129, 2008.

In the next section, we see a more practical, in fact a very popular, API algorithm based on the idea of looking at the fixed-point.

3.3.2 Least-Squares Temporal Difference (LSTD) Learning

The idea here is when it is hard to approximate the fixed-point of the Bellman operator $\mathcal{T}^\pi$ (see Section 3.3.1), the next target would be to approximate the fixed-point of the joint operator $\Pi_? \mathcal{T}^\pi$, where $\Pi_?$ is the projection into a function space w.r.t. norm $\ell_?$. Since $\mathcal{T}^\pi$ is a contraction, if $\Pi_?$ is non-expansive, then the joint operator $\Pi_? \mathcal{T}^\pi$ is a contraction, and thus, has a unique fixed-point (our approach has a unique solution).

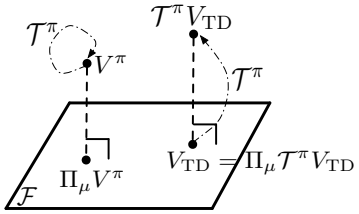
$\Pi_\infty \mathcal{T}^\pi$ is a contraction in ℓ_∞-norm, but ℓ_∞-projection is numerically expensive when the number of states is large or infinite.

LSTD searches for the fixed-point of $\Pi_{2,\mu} \mathcal{T}^\pi$, where $\Pi_{2,\mu}$ is defined as

$$\Pi_{2,\mu} g = \arg\min_{f \in \mathcal{F}} \|f - g\|_{2,\mu}.$$

When the fixed-point of $\Pi_{2,\mu} \mathcal{T}^\pi$ exists, we call it the LSTD solution, i.e.,

$$V_{\mathrm{TD}} = \Pi_\mu \mathcal{T}^\pi V_{\mathrm{TD}}.$$



If we assume that the function space $\mathcal{F}$ is the linear space spanned by d feature vectors $\{\varphi_i\}_{i=1}^d$, $\varphi_i : \mathcal{X} \to \mathbb{R}$, i.e., $\mathcal{F} = \{f_\theta \mid \theta \in \mathbb{R}^d \text{ and } f_\theta(\cdot) = \phi(\cdot)^\top \theta\}$, where $\phi(\cdot) = \big(\varphi_1(\cdot), \dots, \varphi_d(\cdot)\big)^\top$ is the feature vector.

From the definition of V_{TD} and $\Pi_{2,\mu}$, we may write

$$\begin{aligned}
&\langle \mathcal{T}^\pi V_{\text{TD}} - V_{\text{TD}}, \varphi_i \rangle_\mu = 0 \qquad\qquad i = 1, \dots, d \\
&\langle r^\pi + \gamma P^\pi V_{\text{TD}} - V_{\text{TD}}, \varphi_i \rangle_\mu = 0 \\
&\underbrace{\langle r^\pi, \varphi_i \rangle_\mu}_{b_i} - \sum_{j=1}^d \underbrace{\langle \varphi_j - \gamma P^\pi \varphi_j, \varphi_i \rangle_\mu}_{A_{ij}} \cdot \theta_{\text{TD}}^{(j)} = 0,
\end{aligned}$$

which means that the LSTD solution $V_{\text{TD}}(\cdot) = \phi(\cdot)^\top \theta_{\text{TD}}$ is obtained as the solution of the linear system of equations $A\theta_{\text{TD}} = b$.

In general, $\Pi_\mu \mathcal{T}^\pi$ is not a contraction and does not have a fixed-point, and thus, LSTD does not have a solution. However, if $\mu = d^\pi$, the stationary distribution of policy π, then $\Pi_\mu \mathcal{T}^\pi$ is a contraction and has a unique fixed-point, and thus, LSTD solution exists.

Now the next question is whether the LSTD solution V_{TD} is good or not?

Theorem 5.

$$\|V^\pi - V_{TD}\|_{d^\pi} \le \frac{1}{\sqrt{1-\gamma^2}} \min_{f \in \mathcal{F}} \|V^\pi - V\|_{d^\pi}.$$

Proof. From the Pythagorean theorem, we have (see the above figure)

$$\begin{aligned}
\|V^\pi - V_{\text{TD}}\|_{d^\pi}^2 &= \|V^\pi - \Pi_{d^\pi} V^\pi\|_{d^\pi}^2 + \|V_{\text{TD}} - \Pi_{d^\pi} V^\pi\|_{d^\pi}^2 \\
&= \|V^\pi - \Pi_{d^\pi} V^\pi\|_{d^\pi}^2 + \|\Pi_{d^\pi} \mathcal{T}^\pi V_{\text{TD}} - \Pi_{d^\pi} \mathcal{T}^\pi V^\pi\|_{d^\pi}^2 \\
&\le \|V^\pi - \Pi_{d^\pi} V^\pi\|_{d^\pi}^2 + \gamma^2 \|V^\pi - V_{\text{TD}}\|_{d^\pi}^2.
\end{aligned}$$

The proof follows immediately from the last inequality. □

Algorithm

1. We build a data set of the form $\mathcal{D} = \big\{\big(x_i, \pi(x_i), r_i = r^\pi(x_i), x_i'\big)\big\}_{i=1}^N, \quad x_i \sim d^\pi$.
2. From this data set, we compute

$$\widehat{A}_{ij} = \frac{1}{N} \sum_{l=1}^N \varphi_i(x_l)\big[\varphi_j(x_l) - \gamma \varphi_j(x_l')\big], \qquad \widehat{b}_i = \frac{1}{N} \sum_{l=1}^N \varphi_i(x_l) r_l \,.$$

3. We obtain $\widehat{\theta}_{\text{TD}}$ as the solution of the linear system of equation $\widehat{A}\widehat{\theta}_{\text{TD}} = \widehat{b}$ (if the solution exists).

When $N \to \infty$ then $\widehat{A} \to A$ and $\widehat{b} \to b$, and thus, $\widehat{\theta}_{\text{TD}} \to \theta_{\text{TD}}$ and $\widehat{V}_{\text{TD}} \to V_{\text{TD}}$.

This algorithm can be extended to control (full RL) by moving from V function to Q function and adding the greedification process after each LSTD-style policy evaluation. The resulting algorithm is called least-squares policy iteration (LSPI).

For LSTD algorithm, see ***Reference:***

- S. Bradtke and A. Barto. Linear least-squares algorithms for temporal difference learning. Machine Learning 22:33?57, 1996.

For LSPI algorithm, see ***Reference:***

- M. Lagoudakis and R. Parr. Least-squares policy iteration. Journal of Machine Learning Research, 4:1107-1149, 2003.

For the full analysis of LSTD and LSPI, see ***Reference:***

- A. Lazaric, M. Ghavamzadeh, and R. Munos. Finite-Sample Analysis of Least-Squares Policy Iteration. Journal of Machine Learning Research (JMLR), 13:3041-3074, 2012.