STAT 234 – Sequential Decision Making March 24, 2021

Lecture : Least-Squares Methods in RL

Instructor: Susan Murphy *Scribe: Kelly Zhang*

Contents

1 Overview and Problem Setup **-1**

2 Generalized Policy Iteration **-1**

3 Least Squares Temporal Difference Learning Q (LSTD-Q) **-2**
3.1 Overview . . . . . . -2
3.2 Bellman Residual Minimization . . . . . . -3
3.3 Fixed Point Approximation . . . . . . -5
3.4 Projection Step . . . . . . -5

4 Least-Squares Policy Iteration (LSPI) **-6**
4.1 Comparison to Batch Q-Learning . . . . . . -7

1 Overview and Problem Setup

The focus of this lecture is to better understand Least-Squares Policy Iteration and the paper by Lagoudakis and Parr (`https://www2.cs.duke.edu/research/AI/LSPI/jmlr03.pdf`). Some other resources on batch off policy learning are a tutorial on offline RL by Levine et al. (`https://arxiv.org/pdf/2005.01643.pdf`) and a review of batch RL by Lange et al. (`http://tgabel.de/cms/fileadmin/user_upload/documents/Lange_Gabel_EtAl_RL-Book-12.pdf`).

Throughout, we assume that we are in the off-policy setting, in which we have access to data collected with a potentially unknown policy. In other words, we have access to data of the form $\mathcal{D}_L = (S_i, A_i, R_i, S_i')_{i=1}^L$, which is collected by some policy. **Our goal is to use this data to estimate an optimal policy, which maximizes sum of discounted rewards.**

Note, as we discussed in our discussion groups, Lagoudakis and Parr do not state explicit assumptions on μ_D, the distribution generating the batch data $\mathcal{D}_L = (S_i, A_i, R_i, S_i')_{i=1}^L$. However, to learn the optimal policy as the number of data tuples goes to infinity the data must satisfy certain assumptions one of which is that the *behavior policy* (the policy that was used to select the actions, A_i's and thus result in the data $\mathcal{D}_L = (S_i, A_i, R_i, S_i')_{i=1}^L$) results in exploration of the entire action space for each state infinitely often.

2 Generalized Policy Iteration

Generalized Policy Iteration (the authors also call this an Actor-Critic architecture) is an optimization method to use data to estimate an optimal policy. Policy iteration involves two steps, which we iterate between:

1. Policy Evaluation (Learning Algorithm)

2. Policy Improvement (Action Selection Strategy)

Policy Evaluation Policy evaluation takes a policy π, and constructs an estimate of the Q-function for π. Recall that the Q-function for policy π gives us the expected reward for following that policy π given each possible state and action:

$$Q^\pi(s,a) = E_\pi\left[\sum_{s=t}^{\infty} \gamma^{s-t} R_t \middle| S_t = s, A_t = a\right]$$

In other words, policy evaluation methods takes a policy π and data $\mathcal{D}_L = (S_i, A_i, R_i, S_i')_{i=1}^L$ and outputs $\hat{Q}^\pi$. [1]

Policy Improvement Policy improvement takes a Q-function for some policy π and estimates a policy $\hat{\pi}$ that has even greater value than π. For example, given some Q-function Q^π, we can find improved policy $\hat{\pi}$, by choosing the policy that acts greedily with respect to Q^π, i.e., we can set $\hat{\pi}(s) = \text{argmax}_{a\in\mathcal{A}} Q^\pi(s,a)$. Overall, policy improvement takes a Q-function for some policy π and outputs a improved policy $\hat{\pi}$.

Generalized Policy Iteration Generalized policy iteration takes data $\mathcal{D}_L = (S_i, A_i, R_i, S_i')_{i=1}^L$ and uses it to estimate an optimal policy by iterating between policy evaluation and policy improvement. Specifically, suppose we start with some initial policy $\hat{\pi}_0$. We use policy evaluation to estimate $\hat{Q}^{\hat{\pi}_0}$. We then use policy improvement to estimate a better policy $\hat{\pi}_1$. We then use policy evaluation to estimate $\hat{Q}^{\hat{\pi}_1}$. Then we can use policy improvement again to estimate a better policy $\hat{\pi}_2$. And so on.

Least-Squares Policy Iteration

- Least Squares Temporal Difference Q (LSTDQ) is a method for policy evaluation (learning Q function for some policy π)
- Least Squares Policy Iteration iterates between policy evaluation, using LSTDQ, and policy improvement, using a greedy policy.
- LSPI uses a linear model to approximate the Q-function. Specifically, it approximates $Q^\pi(s,a)$ with $w^\top \phi(s,a)$, where $\phi(s,a)$ is a basis function.

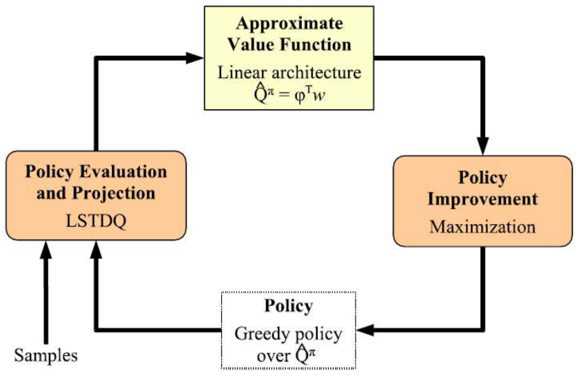
3 Least Squares Temporal Difference Learning Q (LSTD-Q)

3.1 Overview

Given a policy π, we want to learn Q^π. We use a linear model to approximate Q^π:

$$\hat{Q}^\pi = w^\top \phi(s,a)$$

[1]We subscripted the expectation E by the policy π to remind ourselves that the expectation is over the triplet: the policy, transition probability function and reward function. The latter two are specified via the data generating distribution, μ_D.



Approximate Value Function
Linear architecture
$\hat{Q}^\pi = \varphi^\top w$
Policy Evaluation and Projection
LSTDQ
Policy Improvement
Maximization
Policy
Greedy policy over $\hat{Q}^\pi$
Samples

Figure 3: Least-squares policy iteration.

While before in the tabular setting the Q-function was a $|\mathcal{S}| \times |\mathcal{A}|$ sized table, we now assume that the dimensionality of $\phi(s,a)$ is much smaller than $|\mathcal{S}| \cdot |\mathcal{A}|$.

Above $\phi(s,a)$ is a basis function; this function is assumed to be given and is not learned by LSTD-Q. If basis functions $\phi(s,a)$ are chosen poorly the LSTD-Q method may not be able to estimate Q^π well. By estimating Q^π, we mean we want to estimate parameters w. What criterion do we want to optimize in our choice of w?

Recall that by the Bellman equation, we have the following optimality property for the Q-function:

$$\begin{aligned} Q^\pi(s,a) &= E_\pi \left[R_t + \gamma Q^\pi(S_{t+1}, A_{t+1}) \middle| S_t = s, A_t = a \right] \\ &= E\left[R_t + \gamma \sum_{a' \in \mathcal{A}} \pi(a'|S_{t+1}) Q^\pi(S_{t+1}, a') \middle| S_t = s, A_t = a \right] \end{aligned} \tag{1}$$

Note that if π is a deterministic policy (meaning given a state, it chooses an action with probability 1), we can rewrite the above as $Q^\pi(s,a) = E_\pi \left[R_t + \gamma Q^\pi\left(S_{t+1}, \pi(S_{t+1})\right) \middle| S_t = s, A_t = a \right]$, where $A_{t+1} = \pi(S_{t+1})$. For the rest of this document, for notational clarity, we assume that evaluation policy π is deterministic.

Note that in optimality property defined in equation (1), Q^π is on both sides of the equation. Next, we discuss criterion we can use to estimate w.

3.2 Bellman Residual Minimization

We are interested in estimating a w that parameterizes Q^π for some policy we want to evaluate, π. Given equation (1), using our data $\mathcal{D}_L = (S_i, A_i, R_i, S_i')_{i=1}^L$, we might think a natural way to estimate w is using the following least squares criterion:

$$\hat{w} = \underset{w}{\operatorname{argmin}} \sum_{i=1}^{L} \left(\underbrace{R_i + \gamma w^\top \phi\left(S_i', \pi(S_i')\right)}_{\text{Target}} - w^\top \phi(S_i, A_i) \right)^2 \tag{2}$$

where we have replaced $Q^\pi(s,a)$ with $w^\top \phi(s,a)$. The above criterion is also called "Bellman residual minimization". Note that $\hat{w}$ is an M-estimator because it is the minimizer / maximizer of a criterion function.

Let us examine this criterion further. Taking the derivative of equation (2) with respect to w and setting it equal to zero (to find minimizing value), we get that $\hat{w}$ is such that

$$0 = \sum_{i=1}^{L} \Big(R_i + \gamma \hat{w}^\top \phi\left(S_i', \pi(S_i')\right) - \hat{w}^\top \phi(S_i, A_i) \Big) \left(\gamma \phi\left(S_i', \pi(S_i')\right) - \phi(S_i, A_i) \right) \tag{3}$$

Equation (3) above is an estimating equation for w. As discussed in the handout on M and Z-estimators on canvas, a necessary (but not sufficient) property of M-estimators that are consistent[2], is that when they are differentiable, the derivative has expectation zero when we plug in the true value of w in for $\hat{w}$. Specifically, let us assume for now that there exists some optimal w_0 such that $Q(s,a) = w_0^\top \phi(s,a)$ for all s, a (we relax this assumption later in our discussion of projections). Let us examine the expectation of estimating equation (3) when we plug in w_0:

$$E\left[\sum_{i=1}^{L} \Big(R_i + \gamma w_0^\top \phi\left(S_i', \pi(S_i')\right) - w_0^\top \phi(S_i, A_i) \Big) \left(\gamma \phi\left(S_i', \pi(S_i')\right) - \phi(S_i, A_i) \right) \right]$$

By law of iterated expectations,

$$= \sum_{i=1}^{L} E\left[E\left[\Big(R_i + \gamma w_0^\top \phi\left(S_i', \pi(S_i')\right) - w_0^\top \phi(S_i, A_i) \Big) \left(\gamma \phi\left(S_i', \pi(S_i')\right) - \phi(S_i, A_i) \right) \Big| S_i, A_i \right] \right]$$

$$= \sum_{i=1}^{L} \underbrace{E\left[E\left[\Big(R_i + \gamma w_0^\top \phi\left(S_i', \pi(S_i')\right) - w_0^\top \phi(S_i, A_i) \Big) \gamma \phi\left(S_i', \pi(S_i')\right) \Big| S_i, A_i \right] \right]}_{(a)}$$
$$- \underbrace{E\left[\Big\{ E\left[R_i + \gamma w_0^\top \phi\left(S_i', \pi(S_i')\right) \big| S_i, A_i \right] - w_0^\top \phi(S_i, A_i) \Big\} \phi(S_i, A_i) \right]}_{(b)}$$

Note that by the Bellman equation (1) and by our earlier assumption on w_0, we know that $E\left[R_i + \gamma w_0^\top \phi\left(S_i', \pi(S_i')\right) \big| S_i, A_i \right] = w_0^\top \phi(S_i, A_i)$, so term (b) above equals zero. However, there is no reason to expect that $E\left[R_i + \gamma w_0^\top \phi\left(S_i', \pi(S_i')\right) \big| S_i, A_i, S_i' \right] = w_0^\top \phi(S_i, A_i)$ based on the Bellman equation, which means that term (a) doesn't equal zero generally.

Given that the estimating equation (3) does not necessarily have expectation zero when we plug in the true value of w, w_0, we do not expect $\hat{w}$ to be a consistent estimator for w_0. For this reason, the Bellman residual minimizer is a poor choice of estimator. Next, we discuss an improved estimator.

[2] Asymptotically unbiased as $L \to \infty$.

3.3 Fixed Point Approximation

Rather than using M-estimation criterion (3), we use the following Z-estimation criterion to estimate w_0:

$$0 = \sum_{i=1}^{L} \Big(R_i + \gamma \hat{w}^\top \phi\left(S_i', \pi(S_i')\right) - \hat{w}^\top \phi(S_i, A_i) \Big) \phi(S_i, A_i) \tag{4}$$

For $\hat{w}$ to be a consistent estimator, it is necessary for (4) to have expectation zero when we plug in w_0, the optimal value of w (see M and Z estimator document on canvas).

$$E\left[\sum_{i=1}^{L} \Big(R_i + \gamma w_0^\top \phi\left(S_i', \pi(S_i')\right) - w_0^\top \phi(S_i, A_i) \Big) \phi(S_i, A_i) \right]$$

By law of iterated expectations,

$$= \sum_{i=1}^{L} E\Big[E\Big[R_i + \gamma w_0^\top \phi\left(S_i', \pi(S_i')\right) - w_0^\top \phi(S_i, A_i) \Big| S_i, A_i \Big] \phi(S_i, A_i) \Big] = 0$$

The last equality above holds by Bellman's equation (1) and by our earlier assumption on w_0, which together imply that $E\left[R_i + \gamma w_0^\top \phi\left(S_i', \pi(S_i')\right) \big| S_i, A_i \right] = w_0^\top \phi(S_i, A_i)$.

We expect that as the number of data tuples $\mathcal{D}_L = (S_i, A_i, R_i, S_{i+1})_{i=1}^L$ goes to infinity, i.e. $L \to \infty$, we expect that $\hat{Q}^\pi(s,a) = \phi(s,a)^\top \hat{w} \to Q^\pi(s,a)$ where convergence happens in probability. For this reason criterion (4), the fixed point approximation, is a better criterion to use than (3), the Bellman residual minimizer.

3.4 Projection Step

Previously, we said that we assumed that there exists some w_0 such that $Q^\pi(s,a) = \phi(s,a) w_0$. However, often this may not be true. In other words, $Q^\pi \in \mathbb{R}^{|\mathcal{S}| \cdot |\mathcal{A}|}$ does not lie in the vector space spanned by the columns of $\Phi = \left[\phi(s_1, a_1)^\top; \phi(s_1, a_2)^\top; ...; \phi(s_{|\mathcal{S}|}, a_{|\mathcal{A}|})^\top \right]$. What are we estimating if this is the case?

Let us recall that Z-estimation criterion for the fixed point approximation (4) solves the empirical version of the following:

$$0 = E_{\mu_D}\left[\sum_{i=1}^{L} \Big(R_i + \gamma w_0^\top \phi\left(S_i', \pi(S_i')\right) - w_0^\top \phi(S_i, A_i) \Big) \phi(S_i, A_i) \right] \tag{5}$$

where we subscripted the expectation E by μ_D to remind ourselves that this expectation is with respect to the data generating distribution. While before we defined $Q^\pi(s,a) = \phi(s,a)^\top w_0$, in the above estimating equation we can more generally define w_0 to be the value of w_0 which makes the above equality hold. While $Q^\pi(s,a)$ is not equal to $\phi(s,a)^\top w$ for any w, the above defines w_0 to be the best *projected* solution. Note that this projection depends on data generating distribution μ_D. For example, if in the dataset $\mathcal{D}_L$, certain state action pairs appear more often than others, the above criterion will find a w that weights the "errors" of the more frequently appearing state action pairs more heavily than the less frequently than "errors" of the less frequently appearing state action pairs.

We can explicitly write what w_0 is as follows. First we define $b_{\mu_D} := E_{\mu_D}\left[\frac{1}{L}\sum_{i=1}^{L} R_i \phi(S_i, A_i)\right]$ and $A_{\mu_D} := E_{\mu_D}\left[\frac{1}{L}\sum_{i=1}^{L} \left[\phi(S_i, A_i) - \gamma\phi\left(S_i', \pi(S_i')\right)\right]\phi(S_i, A_i)\right]$. We can rewrite equation (5), as follows

$$0 = E_{\mu_D}\left[\frac{1}{L}\sum_{i=1}^{L}\Big(R_i + \gamma w_0^\top \phi\left(S_i', \pi(S_i')\right) - w_0^\top \phi(S_i, A_i)\Big)\phi(S_i, A_i)\right] = b_{\mu_D} - w_0^\top A_{\mu_D}$$

Thus we have that $w_0 = A_{\mu_D}^{-1} b_{\mu_D}$.

Ways to Improve Quality of Estimate $\hat{Q}^\pi$ Given that w_0 is the best projected solution, which may or not give us a good estimate of Q^π, what are ways to improve the quality of estimate Q^π?

1. Use more expressive feature space, i.e., changing basis functions ϕ.
2. Restrict learning of Q^π to policies π that are close to the behavior policy used in the training data.
3. Control the behavior policy.

Regarding the second and third points above, when the behavior policy (policy to collect data $\mathcal{D}_L$) is far from evaluation policy π, we can get a poor estimate of Q^π. This is because μ_D may not be the data generating distribution we are interested in and μ_D determines the projected solution we find. By altering the distribution of μ_D or ensuring that π is close to the behavior policy, we can get a better projected solution.

4 Least-Squares Policy Iteration (LSPI)

Least Squares Policy Iteration iterates between policy evaluation, using LSTDQ, and policy improvement, using a greedy policy.

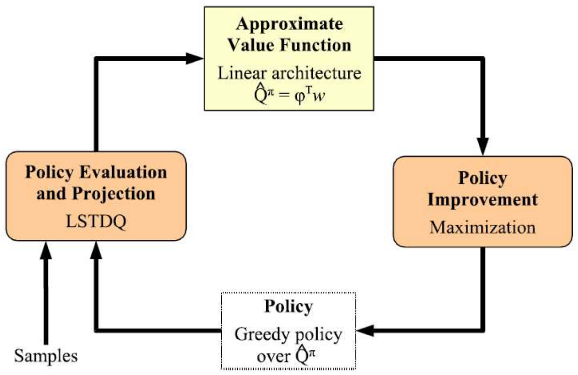

Approximate Value Function
Linear architecture
$\hat{Q}^\pi = \varphi^\top w$
Policy Evaluation and Projection
LSTDQ
Policy Improvement
Maximization
Policy
Greedy policy over $\hat{Q}^\pi$
Samples

Figure 3: Least-squares policy iteration.

Algorithm 1: Least Squares Policy Iteration

```
Inputs: Data tuples $\mathcal{D}_L = (S_i, A_i, R_i, S_i')_{i=1}^L$; basis functions $\phi(s,a) \in \mathbb{R}^k$; discount factor
  $\gamma$; $\epsilon$ stopping criteria; $w_{initial}$ initial weights; $\hat{b} = \frac{1}{L}\sum_{i=1}^L R_i \phi(S_i, A_i)$.
Initialize $w_{new} := w_{initial}$
while $\delta \geq \epsilon$ do
    $w_{old} := w_{new}$
    // (1) Policy Improvement Step
    Define $\pi_{new}$ as the greedy policy over $\hat{Q}^{\pi_{old}}$ (parameterized by $w_{old}$)
    // (2) Policy Evaluation Step (use LSTDQ to estimate $Q^{\pi_{new}}$)
    $\hat{A} := \frac{1}{L}\sum_{i=1}^L \phi(S_i, A_i)\left[\phi(S_i, A_i) - \gamma\phi\left(S_i', \pi_{new}(S_i')\right)\right]^\top$
    $w_{new} := \hat{A}^{-1}\hat{b}$
    // (3) Stopping criteria is if $\hat{Q}^{\pi_{old}}$ and $\hat{Q}^{\pi_{new}}$ are "close"
    $\delta := \|w_{new} - w_{old}\|$
end
Output vector $w_{new} \in \mathbb{R}^k$
```

4.1 Comparison to Batch Q-Learning

First note that a batch version of expected SARSA is in fact batch Q-learning. Recall that in batch Q-learning we are using batch data collected using some unknown algorithm to learn the Q-function for the optimal policy.

Let us define batch Q-learning with linear function approximation. We derive the incremental update for Q-learning by minimizing the following least squares criterion in w:

$$\frac{1}{2}\sum_{i=1}^{L}\Big(R_i + \gamma\phi\left(S_i', \pi(S_i')\right) w_{old} - \phi(S_i, A_i)^\top w\Big)^2$$

Taking the derivative of the criterion in w, we get

$$\sum_{i=1}^{L}\Big(R_i + \gamma\phi\left(S_i', \pi(S_i')\right) w_{old} - \phi(S_i, A_i)^\top w\Big)\phi(S_i, A_i)$$

So the incremental, between sample update is as follows:

$$w_{i+1} := w_i + \alpha_i \underbrace{\Big(R_j + \gamma\phi\left(S_i', \pi_i(S_i')\right)^\top w_i - \phi(S_i, A_i)^\top w_i\Big)\phi(S_i, A_i)}_{\text{one sample estimate of derivative}}$$

where α_i is the learning rate and $\pi_i(s) = \text{argmax}_a\, \phi(s,a)^\top w_i$.

Advantages of LSPI over both Batch Q-Learning

1. LSPI avoids stochastic approximation which uses data very inefficiently
2. Avoids learning rate / step size (no tuning parameters in LSTD-Q and LSPI)
3. Generally stable algorithms (in their experience)