# Value Functions

Aaron Lou

June 2018

## 1 Omitting Policy Gradient

If we omit the policy gradient, then we can just consider our $A^\pi(s_t, a_t)$ and take the optimal policy in particular, our policy would be

$$\pi'(a_t \mid s_t) = \begin{cases} 1 & \text{if } a_t = \arg\max_{a_t} A^\pi(s_t, a_t) \\ 0 & \text{otherwise} \end{cases}$$

and we fit $A^\pi$ in this way.

## 2 Policy and Value Iteration

### 2.1 Policy Iteration

The above method is called Policy Iteration. It works by

1. Evaluate $A^\pi$
2. Set $\pi \leftarrow \pi'$

and we calculate $A^\pi$ as in actor-critic algorithms with

$$A^\pi(s, a) = r(s, a) + \gamma E(|V^\pi(s')| - V^\pi(s)$$

### 2.2 Dynamic Programming

By assuming our set of states and actions is relatively small (and discrete), then we can enumerate our $V^\pi$ in a table. We use bootstrap update to get

$$V^\pi(s) \leftarrow E_{a\sim\pi(a|s)}[r(s, a) + \gamma E_{s'\sim p(s'|s,a)}[V^\pi(s')]$$

and use this to evaluate $V^\pi(s)$. We can also just calculate the raw values of $Q^\pi$ since it's the same thing as $A^\pi$. Our new value iteration method is

1. Set $Q(s, a) \leftarrow r(s, a) + \gamma E_{s'\sim p(s'|s,a)}[V(s')]$.
2. Set $V(s) \leftarrow \max_a Q(s, a)$

### 2.3 Fitted Value Iteration

We create a neural network to represent $V : S \to \mathbb{R}$ with parameters $\phi$. We represent our loss as

$$\mathcal{L}(\phi) = \frac{1}{2}||V_\phi(s) - \max Q^\pi(s,a)||^2$$

We fit our values and get the new fitted value iteration method

1. Set $y_i \leftarrow \max_{a_i}(r(s_i, a_i) + \gamma E[V_\phi(s_i')])$.
2. Set $\phi \leftarrow \arg\min_\phi \frac{1}{2}\sum_i ||V_\phi(s_i) - y_i||^2$.

# 3 Q methods

### 3.1 Q Iteration

We can fit our $Q$ value, which, as it turns out, doesn't require knowledge of transition dynamics

$$Q^\pi(s,a) \leftarrow r(s,a) + \gamma E_{s \sim p(s'|s,\pi(s))}[Q^\pi(s', \pi(s'))]$$

And our policy iteration updates to

1. Evaluate $Q^\pi(s,a)$
2. Set $\pi \leftarrow \pi'$

### 3.2 Fitted Q Iteration

Our fitted form (using a neural net) iteration algorithm is as follows

1. Set $y_i \leftarrow r(s_i, a_i) + \gamma E[V_\phi(s_i')]$
2. Set $\phi \leftarrow \arg\min_\phi \frac{1}{2}\sum_i ||Q_\phi(s_i, a_i) - y_i||^2$

Where we can approximate our value of $E[V_\phi(s_i')]$ with $\max_{a'} Q_\phi(s_i', a_i')$. Although this is a good off-policy method, it doesn't theoretically converge. The full algorithm is listed below

1. Collect dataset $\{(s_i, a_i, s_i', r_i)\}$ using some policy
2. Set $y_i \leftarrow r(s_i, a_i) + \gamma \max_{a_i'} Q_\phi(s_i', a_i')$
3. Set $\phi \leftarrow \arg\max_\phi \frac{1}{2}\sum_i ||Q_\phi(s_i, a_i) - y_i||^2$

where our parameters are, respectively,

1. $N$, our data set size.
2. $K$ the number of iterations (we repeat steps 2 and 3 K times).
3. $S$ the number of gradient steps.

This method is off-policy as step 1 is looking for any values, and our iteration doesn't depend on our current policy.

Our value $\varepsilon$ is our error, and we have

$$\varepsilon = \frac{1}{2} E_{(s,a)\sim\beta}[Q_\phi(s,a) - [r(s,a) + \gamma \max Q_\phi(s',a')]]$$

If $\varepsilon = 0$ then $Q_\phi(s,a) = r(s,a) + \gamma \max_{a'} Q_\phi(s',a')$. This is an optimal Q function, which corresponds to an optimal policy $\pi'$.

### 3.3 Online Q-learning algorithms

We can update our $Q$ iteration algorithm as follows

1. Take some action $a_i$ and observe $(s_i, a_i, s_i', r_i)$.
2. $y_i = r(s_i, a_i) + \gamma \max_{a_i'} Q_\phi(s_i', a_i')$
3. $\phi \leftarrow \phi - \alpha \frac{dQ_\phi}{d\phi}(s_i, a_i)(Q_\phi(s_i, a_i) - y_i)$

In practice, we don't want to have the characteristic function as our first step, as this would remove all other possibilities. Instead, we use

$$\pi(a_t \mid s_t) = \begin{cases} 1-\epsilon \\ \frac{\epsilon}{|\mathcal{A}|-1} \end{cases} \quad \text{"epsilon greedy"}$$

$$\pi(a_t \mid s_t) \propto \exp(Q_\phi(s_t, a_t)) \text{ "Boltzmann exploration"}$$

## 4 Value Function Learning Theory

We define an operation $\mathcal{B} : \mathcal{B}V = \max_a r_a + \gamma \mathcal{T}_a V$. This is our update on $V(s)$ in our value iteration method. We notice there exists a best value $V^*$ such that $BV^* = V^*$. In particular

$$V^*(s) = r(s,a) + \gamma E[V^*(s')]$$

We also know that $\mathcal{B}$ is a contraction (can be proved), so

$$||BV - B\overline{V}||_\infty \leq \gamma ||V - \overline{V}||_\infty \implies ||BV - BV^*||_\infty \leq \gamma ||V - V^*||_\infty$$

so $V \leftarrow BV$ goes to $V^*$. Furthermore, the gap always gets smaller by $\gamma$.

However, in the non-tabular case, our function approximator doesn't work out. In fact, if we reexamine our algorithm, we have

1. Set $y_i \leftarrow \max_{a_i}(r(s_i, a_i) + \gamma E[V_\phi(s_i')])$.
2. Set $\phi \leftarrow \arg\min_\phi \frac{1}{2} \sum_i ||V_\phi(s_i) - y_i||^2$.

We notice $\Pi V = \arg\min_{V' \in \Omega} \in \frac{1}{2} \sum ||V^(s) - V(s)||^2$. Notice that $\Pi$ is a contraction of the $l$-2 norm. However, $\Pi\mathcal{B}$ is not a general contraction, so it doesn't necessarily have to go to an optimal value.

Similarly, for fitted Q-iteration, this isn't a general contraction so it won't converge to our $V^*$. Online Q-learning isn't gradient descent since the target value $[r(s_i, a_i) + \gamma \max_{a'} Q_\phi(s_i', a_i')]$ is constantly changing. As a corollary, batch actor-critic algorithm suffers from the same problem.