MS&E338 Reinforcement Learning Lecture 6 - April 18, 2018

Q-Learning Continued and Value Generalization

Lecturer: Ben Van Roy *Scribe: Cem Kalkanli, James Harrison*

1 Q-Learning Continued

We will first outline several features of two previously discussed algorithms.

Algorithm 1 Episodic Q-Learning

```
1: Initialize with some $Q_{0,0}$
2: for l=0,1,2,... do
3:     Observe $s_0$
4:     for t=0,1,...,$\tau$ do
5:         Select $a_t$
6:         Observe $r_{t+1}, s_{t+1}$
7:         $Q_{l,t+1} = \begin{cases} (1-\gamma_l)Q_{l,t}(s,a) + \gamma_l(r_{t+1} \max_{a \in \mathcal{A}} Q_{l,t}(s_{t+1},a)), & \text{if } (s,a) = (s_t,a_t) \\ Q_{l,t}(s,a), & \text{otherwise} \end{cases}$
8:     end for
9:     $Q_{l+1,0} = Q_{l,\tau}$
10: end for
```

First of all, the action selection procedure can be from any policy. However to ensure convergence, we need to follow a policy that visits any state-action pair infinitely often and picks viable γ_l coefficients.

Aside from that, this is an incremental algorithm meaning that we update each state-action pair as we observe them live. Although this may not be as data as efficient as the following algorithms, compute time for each update does not grow linearly with increasing sample size.

In contrast, we will see a batch algorithm.

Algorithm 2 Batch Sample-Based Value Iteration

```
1: Data:  $\mathcal{D}$={$(s_k, a_k, r_k, s'_k)$:  k=1,2,...,K}
2:        $\mathcal{D}_{s,a}$={$(\bar{s}, \bar{a}, \bar{r}, \bar{s}') \in \mathcal{D}$:  $(\bar{s}, \bar{a}) = (s,a)$}
3:        $K_{s,a} = |\mathcal{D}_{s,a}|$
4: $(\hat{F}Q)(s,a) = \frac{1}{K_{s,a}} \sum_{(s,a,r,s') \in \mathcal{D}_{s,a}} (r + \max_{a' \in \mathcal{A}} Q(s',a'))$
5: $Q(s,a) = \frac{1}{K_{s,a}+\bar{K}}(K_{s,a}(\hat{F}Q)(s,a) + \bar{K}\bar{Q}(s,a))$
```

Let's go over the algorithm. The first major difference from algorithm 1 is that we have accumulated K observations and we are going to run the value iteration on whole data rather than one sample at a time. In this algorithm $\hat{F}$ is our approximate Bellman operator. It is obvious that $\hat{F}$ makes use of every observation we made. Secondly in the update rule, we make use of the terms $\bar{K}$ and $\bar{Q}$, which are fixed regularization parameters. For example we may have some prior knowledge that we want to use about the system. Thus we can deploy them in the update rule. These priors can also help mitigate the effect of noisy observations or small sample sizes. Thus having a regularization term could alleviate some these effects. Note that Episodic Q-learning has natural regularization via $Q_{0,0}$.

Aside from what we have mentioned, compared to the Episodic Q-Learning algorithm, Batch Sample-Based Value iteration is more data efficient. This is due to the fact that the approximate Bellman operator makes use of every observation. However with an increasing number of samples, this method can be cumbersome. This brings us to the following algorithm.

Algorithm 3 Experience Replay

```
1: Buffer:  $\mathcal{D}$
2:            Single-Sample update
3:              sample $(s,a,r,s')$~$Unif(\mathcal{D})$
4:              $Q(s,a) = (1-\gamma)Q(s,a) + \gamma(r + \max_{a' \in \mathcal{A}} Q(s',a'))$
5:          (or)
6:            Sample mini-batch $\tilde{\mathcal{D}} \subset \mathcal{D}$
7:              $Q(s,a) = (1-\gamma)Q(s,a) + \gamma \frac{1}{\tilde{\mathcal{D}}_{s,a}} \sum_{(s,a,r,s') \in \tilde{\mathcal{D}}_{s,a}} (r + \max_{a' \in \mathcal{A}} Q(s',a'))$
```

With Experience Replay algorithm, we have a buffer that contains a fixed number of samples. We can then employ one of two different update rules. By Single-Sample update, we uniformly sample an observation from the buffer and update $Q(s,a)$ accordingly. However with Sample mini-batch method, we draw a subset from the sample set and employ the approximate Bellman operator on that subset.

One of the advantages with this method is that the compute time stays relatively the same even though we can use an update rule similar to the batch algorithm in Algorithm 2. Generally, the batch or single-sample update rules may be applied interchangeably or in combination.

Although these 3 algorithms are known to converge, one of the main problems with these tabular representations is that we can have a very large state space. Because of this constraint, storing $Q(s,a)$ in a tabular form can be very expensive. To overcome this, we will introduce a new notion.

2 Value Generalization

The main goal of the Value Generalization is to allow inference about the states that we did not observe. Rather than trying to update each state-action pair as we observe them, we will parametrize $Q(s,a)$ so that even when the dimensions are very large, non-observed states will be updated. Another name for the Value Generalization is Value function approximation.

As an example, assume that we are trying to learn the mapping from variable x to y and lets say x takes values in $\{0,1\}^N$. As N gets bigger, the cardinality of the domain will grow exponentially. One approach to estimate this function would be to take averages of the output for each x input. But due to the high dimensionality, we may not observe some of the inputs. This example gives an intuition into why we use Value Generalization.

Another example is Tetris.

As you can see from Fig 1, there is an extremely large set of combinations of stationary and moving blocks. Because of this, storing all these states and trying to update them simply won't be feasible for practical purposes.

Motivated by the previous examples, we will try to generalize our observations. One way to achieve this is to parametrize the Q function.

Example 1. *Let's take $\theta \in \mathbb{R}^K$. Then we can parametrize $Q(s,a)$ as $Q_\theta(s,a)$, where*

$$Q_\theta(s,a) = \sum_{k=1}^{K} \theta_k \phi_k(s,a) \tag{1}$$

and ϕ_k's are known functions (usually referred to as features).

In Example 1, we represented Q as a linear model. Another example would a neural network with inputs (s,a) and outputs $Q(s,a)$. Then θ in the previous example would be the weights in the neural network. This approach is usually referred to as "deep" reinforcement learning .

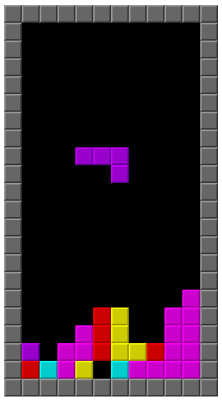

Figure 1: A game of Tetris.

In the rest of this section, we will use linear Q function approximation.

Normally, we use the value iteration update as follows:

$$Q = FQ \tag{2}$$

and we know that theoretically this converges to the optimal value function. For the linearly parametrized Q function, we will have the Approximate Value Iteration:

$$Q = \Pi FQ \tag{3}$$

where $\Pi : \mathbb{R}^{|S| \times |\mathcal{A}|} \to \{Q_\theta : \theta \in \mathbb{R}^K\}$ is a projection onto the K-dimensional subspace. Because of this linear form, each step may be computed efficiently. This update first maximizes the Q function with the Bellman operator, then projects. However, this operation may not always be the optimal one as we will see later.

So now let us define the operation of the projection matrix Π:

$$\Pi Q = \underset{Q_\theta : \theta \in \mathbb{R}^K}{\arg\min} ||Q - Q_\theta||_{2,\mu} \tag{4}$$

where $||Q||_{2,\mu} = (\sum_{s \in S, a \in \mathcal{A}} \mu(s,a) Q^2_{(s,a)})^{\frac{1}{2}}$. Although computationally this is an advantageous representation, there are examples where Approximate VI does not converge.

To prove that this representation and projection does not necessarily converge, assume for an MDP (given below) that for each state $s \in S$, there is only one available action. Then we can replace $Q_{s,a}$ with $V(s)$. With this new notation, we will have the following familiar update rules:

$$V = FV \tag{5}$$

$$V = \Pi FV \tag{6}$$

for VI and approximate VI algorithms. With these updates in mind, we concretize an example that results in non-contraction of ΠF:

Example 2. *We have the setup as in Fig 2. At each state, we have only one action so that we can ignore state-action formulation and think this as a Markov process. For some $\alpha \in (0, 1)$ and $\epsilon > 0$, the transition probabilities are given on the figure. For this setup, we have 0 rewards for each state. Show that approximate VI can diverge from the optimal value function (which is zero for both states).*

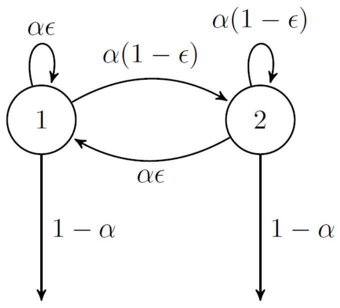

Figure 2: Shows the Markov process in Example 2.

Let us parameterize the value function as $V_\theta(s) = \theta s$ for any $s \in \{1, 2\}$. Then we will have the following s vs $V_\theta(s)$ graph for some $\theta > 0$:

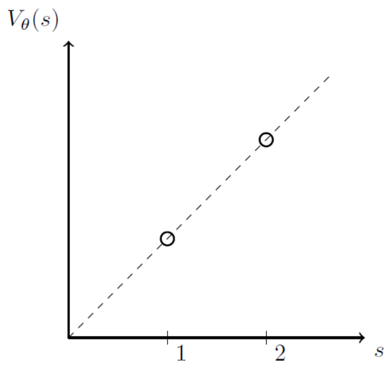

Figure 3: A graph of initial value functions for state 1 and 2, where the dotted line denotes the subspace of value functions parameterized by θ.

When we apply the approximate VI for some appropriate α and ϵ values, we can get the situation depicted in the following figure:

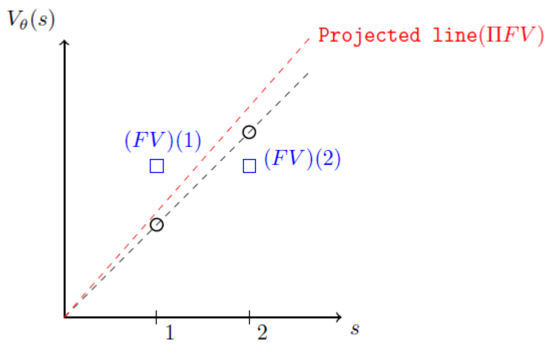

Figure 4: Shows the one iteration of the approximate VI.

We first apply the Bellman operator and get the updated state values as the blue squares. When we project these points onto a 1 dimensional subspace, dashed red line will be our approximate value function. Intuitively we can see this kind of increased slope behavior can lead to a divergent result. Let us now make this observation more rigorous. First apply the Bellman operator to V_θ:

$$\begin{aligned}
(FV)(s) &= \alpha\epsilon V_\theta(1) + \alpha(1-\epsilon)V_\theta(2) && (7)\\
&= \alpha\epsilon\theta + \alpha(1-\epsilon)2\theta && (8)\\
&= \alpha(2-\epsilon)\theta && (9)
\end{aligned}$$

Equation 7 follows from the fact that we have one action per state thus we can ignore the maximization with respect the actions. Also due to the particular nature of the Markov process, Equation 7 is true for all $s \in \{1,2\}$.

Let's take μ's uniform and find the response of the projector Π:

$$\begin{aligned}
\Pi V &= \min_\theta((V_\theta(1) - V(1))^2 + (V_\theta(2) - V(2))^2) && (10)\\
&= \min_\theta((\theta - V(1))^2 + (2\theta - V(2))^2) && (11)
\end{aligned}$$

It is easy to see from Equation 11 that the minimizer θ should be equal to $\frac{V(1)+2V(2)}{5}$. So if follow the approximate VI step by step, we get the following diagram: $V_{\theta_l} \to FV_{\theta_l} \to \Pi FV_{\theta_l} = V_{\theta_{l+1}}$. This means:

$$\begin{aligned}
\theta_{l+1} &= \frac{(FV_{\theta_l})(1) + 2(FV_{\theta_l})(2)}{5} && (12)\\
&= \frac{3}{5}\alpha(2-\epsilon)\theta_l && (13)
\end{aligned}$$

Equation 12 follows from the most recent analysis we did and Equation 13 follows from Equation 9. So if we were to pick $\alpha = 0.99$ and $\epsilon = 0.01$, we would get an increasing sequence of θ's. This would mean that the value function would not converge.

We can visualize this behavior with the following figure.

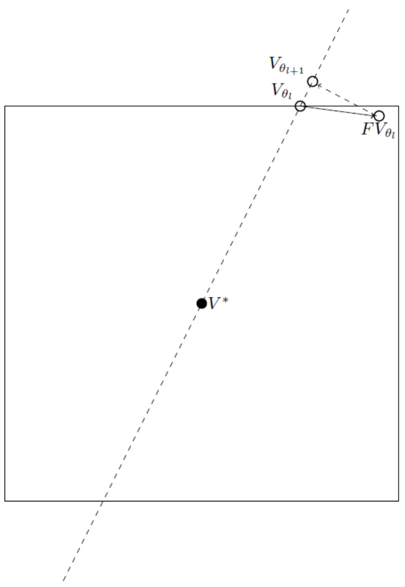

Figure 5: Shows a one iteration of approximate VI for a particular setup. The Bellman operator is a contraction in the infinity norm (the square denotes value functions with infinity norm equivalent to V_{θ_l}), while the projection is in the μ-weighted 2-norm. The dotted line is the subspace of Q functions parameterized by θ.

Now say we have chosen features so that V^* lies on the same line as V_{θ_l}'s. Then we have the situation as in Figure 5 where the dashed line corresponds to that line. Then we apply the Bellman operator first, we are guaranteed to not go further away from V^* with respect to the max norm. But due to the nature of the max norm, we can get FV_{θ_l} as in Figure 5. Then we apply the projection operator, $V_{\theta_{l+1}}$ will be our final destination and as you can see from the figure, we have gone further away from V^* in Euclidean norm (or max norm).

However, although we are not going to prove it, if we were to reformulate the projection operation as follows:

$$\Pi V = \min_{\theta}(\epsilon(V_\theta(1) - V(1))^2 + (1-\epsilon)(V_\theta(2) - V(2))^2) \tag{14}$$

Then we would get a convergent behavior.

Now we will examine a setup where the approximate VI converges. Normally MDPs are defined by $(S, \mathcal{A}, R, P, \rho)$. If we were to have one action per state, we can ignore the action set and reduce the MDP a Markov reward process, written (S, R, P, ρ). Then define $P_{s,s'}$ square matrix as $P_{s,s'} = P_s(s')$ and make the following simplifying assumption:

$$\sum_{s' \in S} P_{s,s'} = \alpha, \quad \forall s \in S \text{ and some } \alpha > 0 \tag{15}$$

Then we can normalize the transition probability matrix P as $\bar{P} = \frac{P}{\alpha}$.

Lemma 1. *If $\mu^T \bar{P} = \mu^T$ for some $\mu \in \mathbb{R}^{|S|}$ such that $\mu \geq 0$ and $1^T\mu = 1$, then $||\bar{P}||_{2,\mu} \leq 1$ where $||\bar{P}||_{2,\mu} = \max_{||V||_{2,\mu}=1} ||PV||_{2,\mu}$.*

Proof. For any V, we have:

$$\begin{aligned}
||\bar{P}V||^2_{2,\mu} &= \sum_{s\in S}\mu(s)(\bar{P}V)^2(s) && (16)\\
&= \sum_{s\in S}\mu(s)(\sum_{s'\in S}\bar{P}_{s,s'}V(s'))^2 && (17)\\
&\leq \sum_{s\in S}\mu(s)\sum_{s'\in S}\bar{P}_{s,s'}V^2(s') && (18)\\
&= \sum_{s'\in S}\sum_{s\in S}\mu(s)\bar{P}_{s,s'}V^2(s') && (19)\\
&= \sum_{s'\in S}\mu(s')V^2(s') && (20)\\
&= ||V||^2_{2,\mu} && (21)
\end{aligned}$$

Equations 16-17 follow from the definition of the weighted norm. Equation 18 follows from Jensen's inequality due to the fact that after the normalization $\{\bar{P}_{s,s'}\}_{s'}$ will represent a probability distribution. Equation 19 is just a switch in the order of the summations. Equation 20 follows from the assumption that $\mu^T\bar{P} = \mu^T$. □

Lemma 2. *$\forall V, V'$ if $\mu^T\bar{P} = \mu^T$ and $\mu \geq 0, 1^T\mu = 1$, then $||FV - FV'||_{2,\mu} \leq \alpha||V - V'||_{2,\mu}$.*

Proof.

$$\begin{aligned}
||FV - FV'||_{2,\mu} &= ||(\bar{R} + PV) - (\bar{R} - PV')||_{2,\mu} && (22)\\
&= ||P(V - V')||_{2,\mu} && (23)\\
&= \alpha||\bar{P}(V - V')||_{2,\mu} && (24)\\
&\leq \alpha||V - V'||_{2,\mu} && (25)
\end{aligned}$$

Equation 22 follows from the fact that we have one action per state thus we can ignore the maximization inside the bellman operator. Equation 24 follows from our initial assumption about P. Equation 25 follows from Lemma 2. □

Theorem 3. $||\Pi||_{2,\mu} \leq 1$ *if* $\Pi V = \arg\min_{\tilde{V}=\Pi\tilde{V}} ||V - \tilde{V}||_{2,\mu}$.

We will not prove Theorem 3 since it is a well known result with regards to the projection operators. Now we will prove the main result of this analysis.

Theorem 4. *If* $\mu^T \bar{P} = \mu^T$ *and* $\mu \geq 0, 1^T\mu = 1$, *then* $\forall V, V'$ $||\Pi FV - \Pi FV'||_{2,\mu} \leq \alpha ||V - V'||_{2,\mu}$.

Proof.

$$\begin{aligned} ||\Pi FV - \Pi FV'||_{2,\mu} &= ||\Pi(FV - FV')||_{2,\mu} && (26)\\ &\leq ||FV - FV'||_{2,\mu} && (27)\\ &\leq \alpha ||V - V'||_{2,\mu} && (28)\end{aligned}$$

Equation 27 follows from Theorem 3. Equation 28 follows from Lemma 2. □

So by Theorem 4, we see a convergent behavior for the approximate VI when certain conditions are satisfied. The figure below shows this update.

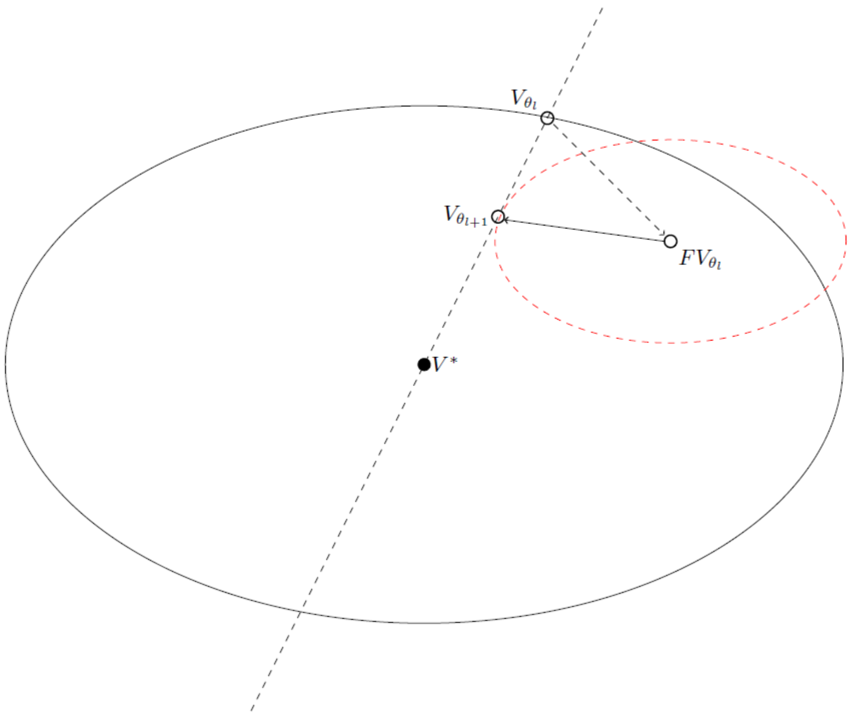

Figure 6: Shows a similar situation as in Fig. 5 except here we use weighted Euclidean norm, which induces the elliptical constraints.