Value Based Functions in Reinforcement Learning

Amir Globerson

1 The Bannach Fixed Point Theorem

Many RL algorithms correspond to a repeated application of some operator. There is a classic result by Bannach that is very useful for showing that such procedures converge to a fixed point.

Definition 1.1 : *Let $(\mathcal{X}, d)$ be some metric space with metric $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Let $\mathcal{T} : \mathcal{X} \to \mathcal{X}$ be an operator from $\mathcal{X}$ to itself. We say that $\mathcal{T}$ is a contraction map if there exists a $0 < c < 1$ such that for all $x, x' \in \mathcal{X}$:*

$$d(\mathcal{T}[x], \mathcal{T}[x']) \leq c d(x, x') \tag{1}$$

In other words $\mathcal{T}$ is a contraction if it "pulls" closer every two points in space. Some metrics of interest to sum will be:

$$\begin{aligned} d_1(\boldsymbol{x}, \boldsymbol{x}') &= \sum_i |x_i - x'_i| \\ d_2(\boldsymbol{x}, \boldsymbol{x}') &= \sqrt{\sum_i (x_i - x'_i)^2} \\ d_\infty(\boldsymbol{x}, \boldsymbol{x}') &= \max_i |x_i - x'_i| \end{aligned}$$

Banach's theorem states that applying $\mathcal{T}$ repeatedly will converge to a fixed point. Consider a sequence $x_0, \ldots, x_t, \ldots$ defined recursively as follows:

$$x_{t+1} = \mathcal{T}[x_t] \tag{2}$$

Then we have the following Bannach's fixed point theorem:

Theorem 1.2 : *[Bannach's Fixed Point Theorem] The sequence $x_0, \ldots, x_t$ converges to a limit x^*, and x^* is a fixed point of $\mathcal{T}$. Namely:*

$$\mathcal{T}[x^*] = x^* \tag{3}$$

Note that this works for any d as long as $\mathcal{T}$ is a contraction with respect to it.

The proof is quite simple, but we won't do all of it. But we can easily see why if a fixed point exists, the algorithm will converge to it. Let's say there exists a point x^* such that $\mathcal{T}[x^*] = x^*$. Then we can easily see that the iterates keep getting closer to x^* because:

$$d(x_{t+1}, x^*) = d(\mathcal{T}[x_t], \mathcal{T}[x^*]) \leq q d(x_t, x^*) \tag{4}$$

And therefore:

$$d(x_{t+1}, x^*) = q^t d(x_0, x^*) \tag{5}$$

To get ϵ close to x^* we therefore need $t = \log \frac{\epsilon}{d(x_0, x^*)}$ iterations. Note this also shows there cannot exist two different fixed points.

2 Value Based Methods

Instead of optimizing the policy directly (as in policy gradient we discussed last time), we can evaluate the impact of actions a in states s and use this to choose the best action given a state. This is the core idea behind methods like value-iteration, Q-learning and more recent deep learning methods like DQN [**?**].

2.1 Value and Q functions for policy π

We begin with some definitions, that you already saw in the previous RL lecture. Given a policy π, the value function $V^\pi(s)$ give the expected discounted reward from starting at state s and using policy π. Namely:

$$V^\pi(s) = \mathbb{E}\left[\sum_t \gamma^t R_t | S_0 = s\right] \tag{6}$$

Above it is implicitly assumed that the expectation is with respect to the distribution over states, actions and rewards corresponding to the MDP and policy π. The following proposition characerizes $V^\pi(s)$ as the solution to a set of linear equations.

Proposition 2.1 : *$V^\pi(s)$ satisfies the following recursive property:*

$$V^\pi(s) = r(s, \pi(s)) + \gamma \sum_{s'} p(s'|s, \pi(s)) V^\pi(s') \tag{7}$$

Furthermore, $V^\pi(s)$ is the unique function $V(s)$ satisfying this set of linear equations.

Proof:

$$\begin{aligned} V^\pi(s) &= \mathbb{E}\left[\sum_t \gamma^t R_t | S_0 = s\right] = \mathbb{E}\left[R_0 | S_0 = s\right] + \mathbb{E}\left[\sum_{t=1} \gamma^t R_t | S_0 = s\right] \\ &= r(s, \pi(s)) + \mathbb{E}\left[\sum_{t=1} \gamma^t R_t | S_0 = s\right] = r(s, \pi(s)) + \mathbb{E}\left[\mathbb{E}\left[\sum_{t=1} \gamma^t R_t | S_0 = s, S_1\right]\right] \end{aligned}$$

where the last transition we use the smoothing theorem to condition over S_1 and then take expectations. Focus on the last expression on the right:

$$\begin{aligned}\mathbb{E}\left[\mathbb{E}\left[\sum_{t=1}\gamma^t R_t|S_0=s,S_1\right]\right] &= \sum_{s'}\mathbb{P}\left[S_1=s'|S_0=s\right]\mathbb{E}\left[\sum_{t=1}\gamma^t R_t|S_0=s,S_1=s'\right] \\ &= \sum_{s'}p(s'|s,\pi(s))\mathbb{E}\left[\sum_{t=1}\gamma^t R_t|S_1=s'\right] \\ &= \gamma\sum_{s'}p(s'|s,\pi(s))\mathbb{E}\left[\sum_{t=1}\gamma^{t-1} R_t|S_1=s'\right]=\gamma\sum_{s'}p(s'|s,\pi(s)V^\pi(s')\end{aligned}$$

where in the last equality we used the fact that the MDP is time invariant so that:

$$\mathbb{E}\left[R_0+\gamma R_1+\gamma^2 R_2+\ldots|S_0=s\right]=\mathbb{E}\left[R_1+\gamma R_2+\gamma^2 R_3+\ldots|S_1=s\right] \quad (8)$$

Putting everything together we have Equation 7. The uniqueness will be shown in Proposition 2.2 below.

□

Another useful measure is the state-action value function, or Q function: $Q^\pi(s,a)$ which gives the expected reward from starting at state s, taking action a, and then following the policy π. Namely:

$$Q^\pi(s,a)=\mathbb{E}\left[\sum_t\gamma^t R_t|S_0=s,A_0=a\right] \quad (9)$$

2.1.1 Calculating V^π and Q^π - Iterative Policy Evaluation

Say you have a given policy π and want to calculate its value function. This could be very useful for finding an improved policy, in algorithms like policy iteration (see below) or its variants.

Here we briefly describe an iterative algorithm, known as **Iterative Policy Evaluation** which converges to V^π. Start with some $V_0(s)$ and obtain V_{t+1} from V_t via:

$$V_{t+1}(s)=r(s,\pi(s))+\gamma\sum_{s'}p(s'|s,\pi(s))V_t(s') \quad (10)$$

Do this for all states s.

Proposition 2.2: *The Iterative Policy Evaluation algorithm in Equation 10 converges to $V^\pi(s)$. Furthermore, $V^\pi(s)$ is the only function satisfying Equation 7.*

Proof: To show that the iterates converge to a fixed point we use the Bannach fixed point theorem. To use it we just need to show that the update Equation 10 is contracting. Denote the update by $V_{t+1}=\mathcal{T}[V_t]$. We will show it is a contraction with respect to

the ℓ_∞ distance.

$$\begin{aligned} d(\mathcal{T}[V], \mathcal{T}[\hat{V}]) &= \|\mathcal{T}[V] - \mathcal{T}[\hat{V}]\|_\infty \\ &= \max_s |r(s,\pi(s)) + \gamma \sum_{s'} p(s'|s,\pi(s))V(s') - r(s,\pi(s)) + \gamma \sum_{s'} p(s'|s,\pi(s))\hat{V}(s')| \\ &\leq \gamma \max_s |\sum_{s'} p(s'|s,\pi(s)(V(s') - \hat{V}(s'))| \leq \gamma \max_s \sum_{s'} p(s'|s,\pi(s)|V(s') - \hat{V}(s')| \\ &\leq \gamma \max_s \max_{s'} |V(s') - \hat{V}(s')| = \gamma \|V - \hat{V}\|_\infty \end{aligned}$$

So we have contraction with constant $\gamma < 1$.

We therefore have that the iterations converge to a fixed point of the update $\bar{V}$, and that this fixed point is unique. Because $\bar{V}$ is a fixed point, it satisfies Equation 7 and is therefore equal to V^π. Note that the Banach theorem also gives us the fact that V^π is the unique solution of Equation 7 since it states that the fixed point is unique. □

We note that a similar approach may be used for calculating Equation 9 by iterating it.

2.2 Optimal Value and Q functions

Our key interest is in optimal policies. So we consider the best values of the value functions, when maximizing over policies. For the value function this corresponds to the optimal value function:

$$V^*(s) = \max_\pi V^\pi(s) \tag{11}$$

Although this tells us the best we can do for state s, it doesn't immediately suggest which action to take in state s. In that context the optimal Q function is very useful. Namely:

$$Q^*(s,a) = \max_\pi Q^\pi(s,a) \tag{12}$$

We first note a useful relations between V^* and Q^* in the following propositions.

Proposition 2.3: *The functions V^* and Q^* are related via:*

$$V^*(s) = \max_a Q^*(s,a) \tag{13}$$

Proof: First, we note that in our MDP is time-invariant and therefore the optimal policy is stationary. i.e., we lose nothing by always taking the same action at the same state. For proving our claim, let us for a moment consider non-stationary policies, which at for $S_0 = s$ take action a, but can later follow a policy π that does not take a at state s. Denote such a policy by (a, π). Then we have:

$$V^*(s) = \max_\pi V^\pi(s) = \max_a \max_\pi V^{(a,\pi)}(s) \tag{14}$$

where the second equality follows from the fact that the RHS optimizes over all possible policies that are non-stationary in the first step. We then have, by definition of Q

$$V^*(s) = \max_a \max_\pi Q^\pi(s,a) = \max_a Q^*(s,a) \tag{15}$$

□

This relation suggests a way of finding the optimal policy, as discussed next.

2.3 Optimal policy from optimal V and Q functions

Next, let's see how to find the optimal policy from Q^*. Intuitively, at state s it makes sense to take the action that maximizes $Q(s,a)$. The next proposition states that this is indeed the optimal policy.

Proposition 2.4: *Suppose that V^* satisfies the Bellman optimality equations in Equation 19. Let Q^* be the corresponding Q function from Equation 18. Define the policy:*

$$\pi^*(s) = \arg\max_a Q^*(s,a) \tag{16}$$

Then π^* is an optimal policy at all states. Namely: $V^{\pi^*}(s) = V^*(s)$. In other words, whichever state you start at, it's always best to act according to π^*.

Proof: We need to show that $V^*(s)$ solves Equation 7 for $\pi = \pi^*$. Starting from RHS of Equation 7 with V^* and π^* we have:

$$r(s,\pi^*(s)) + \gamma \sum_{s'} p(s'|s,\pi^*(s))V^*(s') = Q^*(s,\pi^*(s)) = V^*(s) \tag{17}$$

as required. Since there is a unique such V^* we have that $V^{\pi^*} = V^*$. □

2.4 The Bellman Optimality Equations

The functions V^*, Q^* satisfy a certain set of equations known as the Bellman optimality equations. To derive these, we first observe a relation between Q^* and V^*.

Proposition 2.5: *For all states s it holds that:*

$$Q^*(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a)V^*(s') \tag{18}$$

Proof: Starting with the definition of Q we have:

$$\begin{aligned} Q^*(s,a) &= \max_\pi \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t | S_0 = s, A_0 = a\right] \\ &= \max_\pi r(s,a) + \sum_{s'} p(s'|s,a)\mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t R_t | S_1 = s'\right] \\ &= r(s,a) + \max_\pi \gamma \sum_{s'} p(s'|s,a)\mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} R_t | S_1 = s'\right] \end{aligned}$$

We have an expression of the form $\max_\pi \sum_{s'} f(s',\pi)$. But we know that $\max_\pi f(s',\pi) = V^*(\pi)$, and we know that the $\arg\max$ of this max is always π^*, regardless of the value of s' (this follows from Equation 2.4 above). We can thus conclude that this is equal to: $\sum_{s'} \max_\pi f(s',\pi) = \sum_{s'} V^*(s')$, establishing the claim. □

Propositions 2.3 and 2.5 above give us the well known Bellman optimality equation for V^*, stated next.

Proposition 2.6: *$V^*(s)$ satisfies the following recursive property:*

$$V^*(s) = \max_a r(s,a) + \gamma \sum_{s'} p(s'|s,a) V^*(s') \tag{19}$$

Furthermore it is the unique function $V(s)$ satisfying the above.

Proof: The fact that the optimal policy satisfies Equation 19 follows from Propositions 2.3 and 2.5. The fact that it unique will follow from the analysis of the Value Iteration Algorithm in Proposition 2.7. □

2.5 Optimal Policy using the Value Iteration Algorithm

To calculate the optimal policy in Section 2.3 we need the optimal V^* or Q^*. Finding V^* is equivalent to finding the unique solution of the Bellman equation Equation 19. As in the policy evaluation case, we can consider turning this equation into an iterative algorithm. Namely, start with some V_0 and update:

$$V_{t+1}(s) = \max_a r(s,a) + \gamma \sum_{s'} p(s'|s,a) V_t(s') \tag{20}$$

and update for all s at each iteration. This is the value iteration algorithm. As we claim next, it converges to V^*.

Proposition 2.7: *The value iteration algorithm converges to V^* as $t \to \infty$.*

Proof: It is sufficient to show that value iteration converges to a unique fixed point $\bar{V}$. Since we know that this fixed point will satisfy the Bellman optimality equations and thus $\bar{V} = V^*$. To show this convergence we again can use the Bannach fixed point theorem. Denote by $\mathcal{T}$ the Value Iteration operator in Equation 20. Namely $V_{t+1} = \mathcal{T}[V_t]$. Let's show that it is contracting with respect to the ℓ_∞ norm. It can be shown, similar to policy evaluation that $\mathcal{T}$ is a γ contraction with respect to the ℓ_∞ norm. □

The downside of value iteration is that it requires knowledge of the MDP model to calculate the update. Indeed this can be replaced with various sampling estimates as we discuss later.

2.6 Optimal Policy using the Policy Iteration Algorithm

An alternative to finding an optimal policy is to start from a policy π_0, and the at iteration t:

- Calculate the value function for policy π_t, namely $V^{\pi_t}(s)$, and corresponding $Q^{\pi_t}(s,a)$
- Set $\pi_{t+1} = \arg\max_a Q^{\pi_t}(s,a)$

It can be shown that this improves the expected return at every iteration, and will converge to the optimal policy in a finite number of iterations (assuming calculating the value function can be done in some fixed time).

2.7 Bellman Equations for Q^*

As we saw above, finding the optimal policy is easy when you have Q^*. Indeed many algorithms work by estimating the function Q^*. One of the most well known of these is the Q learning algorithm, proposed by Watkins in 89. One way of understanding it is by characterizing the set of equations that Q^* must satisfy. These can be obtained from combining Equation 18 and Equation 13, resulting in the following proposition.

Proposition 2.8: *The function Q^* satisfies the following property:*

$$Q^*(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \max_{a'} Q^*(s',a') \tag{21}$$

Furthermore, it is the only function $Q(s,a)$ satisfying the above.

Denote the RHS of Equation 21 by $\mathcal{U}(Q)$. Then the proposition specifies that Q^* is the only function satisfying $Q^* = \mathcal{U}(Q^*)$. If we had the transition distribution $p(s'|s,a)$ we could solve the above set of non-linear equations by iterating $Q_{n+1} = \mathcal{U}(Q_n)$ and this can be shown to converge to Q^*. This algorithm is known as Q-value iteration (it's very similar to Value Iteration which iterates Equation **??** to find V^*). Its updates would be:

$$Q_{t+1}(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \max_{a'} Q_t(s',a') \tag{22}$$

where all s, a pairs should be updated. In the next section we discuss some limitations of this approach.

3 Model Free Policy Optimization and Q Learning

All the methods above made two assumptions. First, that we know the MDP model. Second, that the state space is small enough that we can enumerate over it. Both of these are unlikely to hold in cases of interest. In this section, we address the first difficulty. Namely, that of knowing the model. To overcome it, we will replace taking expectations with sampling from the model.

3.1 Monte Carlo Estimates for Policy Evaluation

Say we want to evaluate the value function $V^\pi(s)$ for a particular policy π. The simplest way of doing this would be just sample many sequences of state and action (chosen by π), evaluate their discounted reward, and average. Given enough such sequences, their average will converge to the true expected reward, and hence the true value function. But this may be too costly. Next, we describe a method that learns with experience, and thus progresses more quickly towards the correct solution.

3.2 Q Learning

The Q learning algorithm is an instance of Temporal-Difference (TD) algorithm, which combines sampling with iterative (aka Dynamic Programming) approaches like value

iteration. The algorithm iteratively updates an estimate of the Q^* function by sampling states and actions. It proceeds as follows:

- Initialize $Q(s, a)$ for all s, a.
- Assume we are at state s. Sample action a according to some policy. Receive reward $r(s, a)$ and let the system transition to state s'.
- Update Q as follows:

$$Q(s,a) \leftarrow Q(s,a) + \alpha_t \left(r(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a) \right) \tag{23}$$

  where α_t is a learning rate that changes over time (needs to decay to zero but not too quickly).

It turns out that this converges to the optimal Q^* under certain conditions, as shown in [**?**].

There are several ways to derive the algorithm. One, discussed below, is that we would like to find a Q solving Equation 21. Rewrite Equation 21 as:

$$Q(s,a) = g(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \max_{a'} Q(s',a') \tag{24}$$

Since $g(s, a)$ involves a sum over s', we approximate it by sampling s', yielding the estimate:

$$\hat{g}(s,a) = r(s,a) + \gamma \max_{a'} Q(s',a') \tag{25}$$

Now consider the function:

$$\ell(Q) = (Q(s,a) - \hat{g}(s,a))^2 \tag{26}$$

Then the Q update is exactly a gradient descent update for $\ell(Q)$. It is not a standard gradient descent algorithm however, since ℓ will be different for each iteration.

3.2.1 Analysis for Deterministic Case

Proving that Q learning converges is non-trivial due to its stochastic nature. Things become easier if the system is deterministic, namely knowing s, a deterministically determines the next step. This is based on notes by Shie Mannor, following a proof by Tom Mitchell. Denote by Q_t the value of Q at iteration t. We would like to analyze its distance from the optimal Q^*. Write:

$$\Delta_t = \|Q_t - Q^*\|_\infty = \max_{s,a} |Q_t(s,a) - Q^*(s,a)|$$

To denote the maximum difference between the two functions over all s, a values. Assume a constant step size of $\alpha = 1$ for the update. Because the transition is deterministic, the Bellman equation Equation 21 becomes:

$$Q^*(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \max_{a'} Q^*(s',a') = r(s,a) + \gamma \max_{a'} Q^*(\gamma(s),a') \tag{27}$$

where $\gamma(s)$ is the state deterministically follows s.

In the Q learning update at time t, we will be in state s_t and take action a_t, and then transition to state $s_{t+1} = \gamma(s_t)$. So we have:

$$\begin{aligned}
|Q_{t+1}(s_t, a_t) - Q^*(s_t, a_t)| &= |r(s_t, a_t) + \gamma \max_{a'} Q_t(s_{t+1}, a') - (r(s_t, a_t) + \gamma \max_{\hat{a}} Q^*(\gamma(s_t), \hat{a}))| \\
&= \gamma |\max_{a'} Q_t(s_{t+1}, a') - \max_{\hat{a}} Q^*(s_{t+1}, \hat{a}))| \\
&\leq \gamma \max_{a} |Q_t(s_{t+1}, a) - Q^*(s_{t+1}, a)| \\
&\leq \gamma \max_{s,a} |Q_t(s, a) - Q^*(s, a)| \leq \gamma \Delta_t
\end{aligned}$$

This means that the entry s_t, a_t in the table Q_{t+1} has improved by a factor γ.

Now assume that between n_1 and n_2 we sampled all pairs (s, a). Then it's simple to conclude that $\Delta_{n_2} \leq \gamma \Delta_{n_1}$, and therefore if we saw all items infinitely often, the algorithm will converge to Q^*.

4 Learning in Large State Spaces - Value Function Approximation

When the state space is large we can no longer keep Q, V in tabular form (namely a value for each s for V or each s, a for Q). It then makes sense to parameterize these functions. For example, for the V case, aprroximate $V^*(s)$ via some function $g(h; \boldsymbol{\theta})$. Here g can be for example a neural net whose input is s (in some vector representation). Similarly for Q^* we can have a function $h(s, a; \boldsymbol{\theta})$. To simplify notation we write $Q_{\boldsymbol{\theta}}(s, a)$ in what follows.

We can then consider the same derivation of Q learning, only rewriting Equation 26 as

$$\ell(\boldsymbol{\theta}) = (Q(s, a; \boldsymbol{\theta}) - \hat{g}(s, a; \boldsymbol{\theta}_t))^2 \tag{28}$$

Namely, we take the target to be the RHS of the Bellman equation for the current θ_t and try to optimize $\boldsymbol{\theta}$ to satisfy the Bellman equations. The resulting update would be:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha_t \left[r(s, a) + \gamma \max_{a'} Q(s', a'; \boldsymbol{\theta}_t) - Q(s, a; \boldsymbol{\theta}_t) \right] \nabla_{\boldsymbol{\theta}} Q(s, a) \tag{29}$$

Note that when $\boldsymbol{\theta}$ has a parameter for each (s, a) pair, this becomes the standard Q learning.

4.1 A Counter Example for Function Approximation

The approach above assumes we can replace the RHS of the Bellman optimality equation with its value at the current parameter estimate θ_t. Let's see a case where this is a bad idea. This is example is from [**?**]. We consider estimating $V^*(s)$ rather than Q. Assume we have a parameter $\boldsymbol{\theta}_t$ at time t. Then our goal is to find $\boldsymbol{\theta}_{t+1}$ to mimimize:

$$\sum_s \left(V(s; \boldsymbol{\theta}) - \max_a \left[r(s, a) + \gamma \sum_{s'} p(s'|s, a) V(s'; \boldsymbol{\theta}_t) \right]) \right)^2 \tag{30}$$

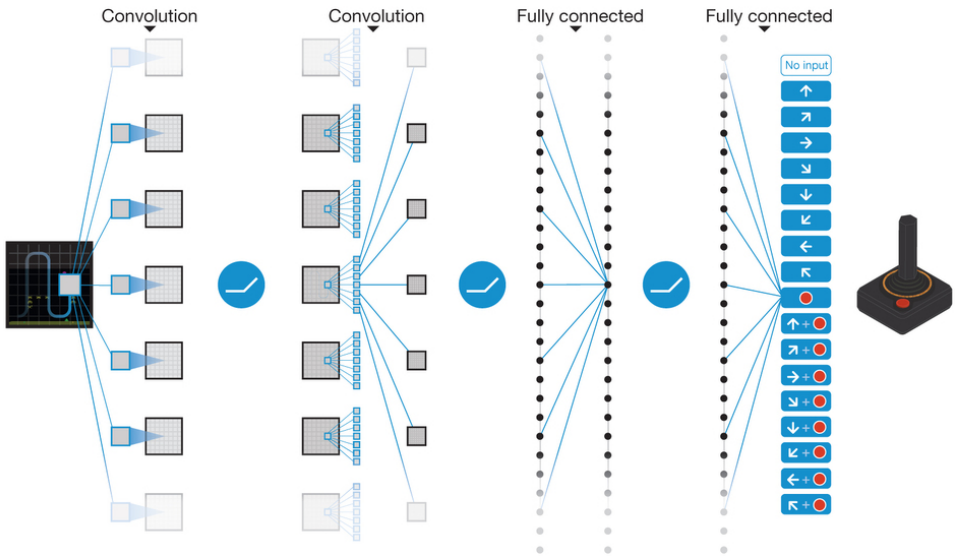

Figure 1: Example of Q function network, taking as input the current game state and outputting values for each of the possible actions. Figure from [**?**].

Now consider a simple case with two states $s = 1, 2$. And transitions not dependent on actions:

$$\begin{aligned} p(2|1, a) &= 1 \\ p(2|2, a) &= 1 \end{aligned}$$

Rewards are always zero. So that the optimal value function is 0. Now assume an approximation function dependent on a single parameter θ as follows:

$$\begin{aligned} V(1; \theta) &= \theta \\ V(2; \theta) &= 2\theta \end{aligned}$$

Note that $\theta = 0$ will give us the correct estimate. Rewriting the objective Equation 30 for this case we get:

$$(\theta - \gamma 2\theta_t)^2 + (2\theta - \gamma 2\theta_t)^2 \tag{31}$$

Differentiating wrt θ and setting to zero yields:

$$\theta_{t+1} = \frac{6}{5}\gamma\theta_t \tag{32}$$

Assuming $\gamma > 5/6$ we have a divergent series of θ_t, which is clearly not a good thing. It can be shown that under additional conditions on the problem, convergence is possible.