Planning via Policy Iteration

Ryan P. Adams*
COS 324 – Elements of Machine Learning
Princeton University

Here we continue to explore MDPs, including an extension of value iteration to an infinite horizon setting and a discussion of the important method of policy iteration.

1 Infinite Horizon Value Iteration

So far, we've focused only the finite horizon case. What about the infinite horizon, total discounted reward case? It turns out that the value iteration idea generalizes quite easily to this case. First, we need to modify the value-iteration equations, defining the k-step-to-go rewards and policies, to take into account the discount factor:

$$\begin{aligned} V_0(s) &= 0 && (1)\\ Q_k(s,a) &= R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s,a) V_{k-1}(s') && (2)\\ \pi^\star_k(s) &= \operatorname*{arg\,max}_{a \in \mathcal{A}} Q_k(s,a) && (3)\\ V_k(s) &= Q_k(s, \pi^\star_k(s)) && (4) \end{aligned}$$

We consider now what happens as we take k to infinity. Does this limit generate a policy $\pi^\star(\cdot)$ that is optimal for the infinite horizon case? This idea is implemented in the infinite horizon version of value iteration, shown in Algorithm 1. This algorithm is very similar to the k-to-go value iteration procedure, except it now iterates on the same set of values, discounting them each time. It loops until the values converge and it produces a single policy.

1.1 Analysis

Does the infinite horizon value iteration algorithm work? Does the value function converge? If it does converge, does it converge to the optimal value function? Intuitively the algorithm makes sense, as it takes the finite horizon case to the limit, but sometimes our intuitions break down going from finite problems to infinite problems. In order to answer these questions, we need some analysis.

*These notes are adapted from Harvard CS181 course notes by Avi Pfeffer and David Parkes.

Algorithm 1 Infinite Horizon Value Iteration

```
function ValueIteration($\gamma$)                                  ▷ Takes a discount factor as input.
    for $s \in \mathcal{S}$ do                                      ▷ Loop over each possible ending state.
        $V(s) \leftarrow 0$                                         ▷ Initialize states with zero value.
    end for
    repeat
        $V_{\text{old}}(\cdot) \leftarrow V(\cdot)$                 ▷ Store off old value function.
        for $s \in \mathcal{S}$ do                                  ▷ Loop over all the states again.
            for $a \in \mathcal{A}$ do                              ▷ Loop over possible actions.
                $Q(s, a) \leftarrow R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s, a) V_{\text{old}}(s')$   ▷ Compute $Q$-function.
            end for
            $\pi^\star(s) \leftarrow \arg\max_{a \in \mathcal{A}} Q(s, a)$   ▷ Find best action from state $s$.
            $V(s) \leftarrow Q(s, \pi^\star(s))$                    ▷ Update value for state $s$.
        end for
    until $|V(s) - V_{\text{old}}(s)| < \epsilon, \ \forall s \in \mathcal{S}$   ▷ Loop until convergence for some small $\epsilon$.
    Return $\pi^\star(\cdot)$
end function
```

The first concept to define is that of a *stationary* policy. A stationary policy is one that only depends on the current state, and not on time or history. A stationary policy $\pi(\cdot)$ will always specify the same action $\pi(s)$ at a state s. (We continue to consider only deterministic policies.)

It is a fact that in the infinite horizon case, there is an optimal stationary policy. Intuitively this is clear. No matter what has happened in the past, when we are at a state s we need to choose the action that maximizes our expected utility from here on out. And the future from state s looks the same any time we are in state s because the horizon is infinite. This is in contrast to the finite horizon case. With a finite horizon, the policy depends on the horizon. If the horizon is small, a greedy policy that optimizes immediate reward will be better. If the horizon is large, a long-term view makes more sense.

For the infinite horizon case, since we know that there exists an optimal stationary (and deterministic) policy, we can restrict our attention to such policies. The next question to ask is, suppose we play a stationary policy $\pi(\cdot)$. What is the expected utility of beginning at a state s, and playing the policy $\pi(\cdot)$? This is called the *value of s under* π, or the *MDP value of* π, and is denoted by $V^\pi(s)$.

We can answer this question by the following observation. Since π is stationary, we know that if we reach a state s' after one step, the expected future reward starting from s' will be $V^\pi(s')$. Furthermore, π specifies what action to take at s, so we know the distribution over the successor states s'. Therefore, the expected reward from s under π is given by

$$V^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s, \pi(s)) V^\pi(s') \tag{5}$$

We have an equation like this for each state s. These equations define a set of N (for N states) linear equations with N variables. It can be shown that they in fact have a unique solution.

So, where are we? We know that there is a stationary optimal policy $\pi^\star(\cdot)$. We also know that for any stationary policy, the value function under that policy satisfies Equation 5. So we can plug in $\pi^\star$ to get

$$V^{\pi^\star}(s) = R(s, \pi^\star(s)) + \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s, \pi^*(s)) V^{\pi^\star}(s') \tag{6}$$

But in fact we know something more. We know that $\pi^\star$ is optimal. This implies that $\pi^\star$ specifies the best possible immediate action to take, given that you know you will get $V^{\pi^\star}$ starting at the next state. It follows that

$$\pi^\star(s) = \arg\max_{a \in \mathcal{A}} \left[R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s, a) V^{\pi^\star}(s') \right] \tag{7}$$

and therefore that

$$V^{\pi^\star}(s) = \max_{a \in \mathcal{A}} \left[R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s, a) V^{\pi^\star}(s') \right] \tag{8}$$

Equation 8 specifies a set of equations, known as the *optimality equations*, or the *Bellman equations*.[1] We know that a stationary optimal policy must satisfy these equations. The equations are non-linear (they include a maximization operator), so it is natural to ask whether or not they have a solution, and whether or not the solution is unique.

Theorem 1 *For an MDP with a finite state and action space*

1. *The Bellman equations have a unique solution.*
2. *The values produced by value iteration converge to the solution of the Bellman equations.*

A complete proof of this theorem is beyond the scope of this course. It rests on a very useful tool from real analysis called the **Banach Fixed Point Theorem**, also known as the Contraction Mapping Theorem. Let's try to gain some intuition as to why Theorem 1 is true. We adopt shorthand $V^\star(s) = V^{\pi^\star}(s)$.

Equations of the form of Equation 8 are called *fixed point equations*. The reason is as follows. Suppose we start with some arbitrary value function $V(\cdot)$. We can apply the right hand side of Equation 8 to $V(\cdot)$ to get a new value function $V'(\cdot)$. Thus the right hand side of Equation 8 defines an operator $\mathbb{B}$ on the space of value functions. Equation 8 says that for the optimal value function $V^\star(\cdot)$, we have:

$$V^\star = \mathbb{B}(V^\star) \tag{9}$$

[1] Named after Richard Bellman, who did pioneering work on MDPs in the 1960s.

We call $V^\star$ a *fixed point* of the function $\mathbb{B}$, because $\mathbb{B}$ "fixes $V^\star$ in its place". Now, a natural process to find a fixpoint of $\mathbb{B}$ is as follows. Begin with any value function V_0. Apply $\mathbb{B}$ to it to get V_1. Apply $\mathbb{B}$ again to get V_2, and so on. This is in fact exactly what value iteration does! We have

$$\begin{aligned}
&\text{initialize } V_0(s) = 0 \text{ for all } s \\
&V_1 \leftarrow \mathbb{B}(V_0) \\
&V_2 \leftarrow \mathbb{B}(V_1) \\
&V_3 \leftarrow \mathbb{B}(V_2) \\
&\ldots
\end{aligned}$$

The questions we asked at the beginning of this section can now be refined: Does this process of iterating $\mathbb{B}$ converge to a limit? Is the limit unique? The answer is yes, because $\mathbb{B}$ has a special property: it is a *contraction*. This means that given any two value functions V_1 and V_2, with $V_1 \neq V_2$, then $\mathbb{B}$ brings them closer to each other. Formally, the contraction property that is satisfied by the Bellman operator is:

$$||\mathbb{B}(V_1) - \mathbb{B}(V_2)||_\infty \quad \leq \quad \gamma ||V_1 - V_2||_\infty, \tag{10}$$

where $V_1 \neq V_2$ and the norm is the *max norm* (or sup norm), i.e.,

$$||V||_\infty = \max_{s \in \mathcal{S}} |V(s)| \,. \tag{11}$$

The constant $\gamma \in [0, 1)$ is the discount factor. Since $\gamma < 1$, the norm between two distinct points strictly decreases.

Intuitively, if we have a process that keeps bringing points closer to each other, and we iterate the process, then we will converge to a fixed point. The *Contraction Mapping Theorem* says that under appropriate conditions, which are satisfied here, this is in fact the case: the process of iterating $\mathbb{B}$ has a limit, which is the unique fixed point of $\mathbb{B}$.

Example: The function $f(x) = x/2$ has a contraction property, so that $x' \leftarrow x/2$ is a contraction operator, and has unique fixed point $x^\star = 0$.

When an operator is a contraction then we always have a **unique fixed point**. Recall that the contraction property insists that there exist some $\alpha \in [0, 1)$ such that $||f(x) - f(y)|| \leq \alpha ||x - y||$ for all $x \neq y$. This can be easily seen to hold for $f(x) = x/2$. Now, suppose for contradiction that there are two distinct fixpoints, $x^\star \neq y^\star$, so that $||f(x^\star) - f(y^\star)|| = ||x^\star - y^\star||$ but then this is a contradiction with the contraction property. In addition, iterating also converges to the fixed point. Consider any $x \neq x^\star$. Then $||f(x) - x^\star|| = ||f(x) - f(x^\star)|| < ||x - x^*||$, where the first equality holds by the fact that $x^\star$ is a fixed point and the second inequality by the contraction property.

1.2 Convergence rate of Value Iteration

For value iteration, the smaller γ is, the faster the iteration process converges.[2] So in fact, γ has a crucial impact on the complexity of infinite horizon value iteration. For γ close to 1, convergence

[2] This section is based on the discussion in Russell and Norvig.

will be slow. In fact, if we think about it, this is not surprising. When γ is close to 1, it means that future rewards are only slightly discounted, so the optimal policy needs to take a long-term view – just like a large horizon in the finite horizon case.

Theorem 1 states that value iteration will converge in the limit to the unique solution of the Bellman equations. For any desired error between $V^\star(s)$ and $V(s)$, then there is some finite number of iterations after which this error gap will be closed. In this section, we discuss the impact of the contraction property,

$$||\mathbb{B}(V_1) - \mathbb{B}(V_2)||_\infty \leq \gamma||V_1 - V_2||_\infty \,,$$

for any $V_1 \neq V_2$, on the ability of value iteration (VI) to make rapid progress. We will also consider the impact of a residual error in our approximation on the quality of the policy finally adopted by an agent. We first consider the number of updates that will be required for VI to achieve a particular error bound ϵ. For this, we assume that the reward for any state action pair is bounded, with

$$-R_{\max} \leq R(s,a) \leq R_{\max}, \quad \forall(s,a) \tag{12}$$

By the property of a geometric series, we have

$$-\frac{R_{\max}}{1-\gamma} \leq V^\pi(s) \leq \frac{R_{\max}}{1-\gamma}, \quad \forall\pi, \forall s$$

and in particular, we have

$$||V_0 - V^\star||_\infty \leq \frac{2R_{\max}}{1-\gamma}$$

for any initial value function V_0. One update achieves the following

$$||\mathbb{B}(V) - V^\star||_\infty \leq \gamma||V - V^*||_\infty$$

We wish to find K, the number of iterations, such that

$$\gamma^K \frac{2R_{\max}}{1-\gamma} \leq \epsilon$$

and the final value function is within ϵ of the optimal value function. Solving, we find:

$$K = \left\lceil \frac{\log\left[\frac{2R_{\max}}{\epsilon(1-\gamma)}\right]}{\log\left(\frac{1}{\gamma}\right)} \right\rceil$$

where notation $\lceil x \rceil$ takes the *ceiling* of x, i.e. rounds to the next largest integer.

This shows that the rate of convergence is good with respect to the error, with the number of rounds increasing as $\log(\frac{1}{\epsilon})$. On the other hand we see a weakness with respect to discount factor γ. As this goes towards one, then $\log(\frac{1}{\gamma})$ approaches zero from above, and K grows rapidly.

In addition, the following useful condition can be established, connecting the successive change in valuation function with the error with respect to the optimal value function, with

$$\text{if } ||V_{k+1} - V_k||_\infty < \epsilon\frac{(1-\gamma)}{\gamma} \text{ then } ||V_{k+1} - V^\star||_\infty < \epsilon$$

For this reason, the termination condition in VI is often stated in terms of a check to see whether valuation functions have changed by less than $\epsilon\frac{(1-\gamma)}{\gamma}$.

Now, what really matters to an agent is not the accuracy of the value function but rather the accuracy of the policy induced by a value function. Suppose an agent has an estimate V of $V^\star$ after some number of iterations, how good is the policy that an agent would follow if using value function V? Let π denote the policy that is optimal given V. Then, it can be shown that

$$\text{if } ||V - V^\star||_\infty < \epsilon \text{ then } ||V^{\pi_i} - V^\star||_\infty < 2\epsilon\frac{\gamma}{(1-\gamma)} \tag{13}$$

where V^{π_i} is the *true value* to the agent for operating with policy π_i. Note again that the ability to bound the quality of the policy induced by valuation V depends on γ, with the error bound increasing rapidly as γ approaches one

2 Policy Iteration

Value iteration is one algorithm for solving a planning problem. Another is policy iteration. For this, we express the Bellman equations as

$$\pi^\star(s) = \arg\max_{a\in\mathcal{A}}\left[R(s,a) + \gamma\sum_{s'\in\mathcal{S}} P(s' \mid s,a)V^\star(s')\right]. \tag{14}$$

The *policy iteration* algorithm is based on the observation that we can also view Equation 14 as defining a fixed point equation on the space of stationary policies. Notice that the left hand side, $\pi^\star$, appears on the right hand side. This is precisely the shape of a fixed point equation.

This observation suggests that we can use a similar procedure to value iteration: we begin with any policy π_0, apply the right hand side of Equation 14 to generate a new policy π_1, apply the right hand side again to get π_2, and so on. The process is a little more complicated than for value iteration. In value iteration, applying the right hand side of Equation **??** to any previous value function V is easy. For policy iteration, it is not quite as straightforward to apply the right hand side of Equation 14 to a policy π. We need to get from the policy π to its value function. But we know how to do that, because for a stationary policy, the value function V^π satisfies Equation 5. So, given π, we solve Equation 5, which is a system of linear equations, to obtain V^π. We then apply the right hand side of Equation 14 to get a new policy. That is, we do a step of *policy evaluation* and then a step of *policy improvement*. The policy iteration algorithm is shown in Algorithm 2.

Note that we repeat until the policy stops changing. Policy iteration is a neat idea. Does it actually work? The key steps in the proof are to prove the following two claims:

Algorithm 2 Policy Iteration

```
function PolicyIteration($\gamma$)                                        ▷ Takes a discount factor as input.
    $\pi(\cdot) \leftarrow \pi_0(\cdot)$                                   ▷ Initialize the policy in any way.
    repeat
        $\pi_{\text{old}}(\cdot) \leftarrow \pi(\cdot)$                     ▷ Store off the old policy.
        Solve system: $V(s) = R(s, \pi_{\text{old}}(s)) + \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s, \pi_{\text{old}}(s)) V(s')$ for $V(s)$.
        for $s \in \mathcal{S}$ do                                         ▷ Loop over all states.
            for $a \in \mathcal{A}$ do                                     ▷ Loop over all actions.
                $Q(s, a) \leftarrow R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s, a) V(s')$    ▷ Compute $Q$-function.
            end for
            $\pi(s) \leftarrow \arg\max_{a \in \mathcal{A}} Q(s, a)$        ▷ Update policy to be optimal for current $Q$.
        end for
    until $\pi(s) = \pi_{\text{old}}(s), \ \forall s \in \mathcal{S}$       ▷ Loop until the policy converges.
    Return $\pi(\cdot)$
end function
```

1. If policy iteration returns policy π, then π is an optimal policy.

2. The policies produced by policy iteration get better and better in the sense that $V^{\pi}(s) \geq V^{\pi^{\text{old}}}(s)$ is monotonically nondecreasing from iteration to iteration.

From these one can deduce that policy iteration always terminates with the optimal policy after a finite number of steps. Contrast this with value iteration, which only achieves the optimal value function in the limit (but may achieve the optimal policy earlier.)

3 Comparing VI and PI

How do value iteration and policy iteration compare? It is a fact that policy iteration takes at most as many iterations to reach the optimal policy as value iteration does. In practice, it usually takes far fewer iterations. In policy iteration, the policy always changes every iteration. In contrast, in value iteration, the value function changes every iteration, but the optimal policy relative to that value function, i.e., the optimal policy given that that value function will be achieved in future, may stay the same for several successive iterations.

On the other hand, each individual iteration of policy iteration takes longer, because it requires solving the complete system of linear equations of Equation 5. This can be done via Gaussian elimination, for example, in $O(N^3)$. Compare with value iteration, in which each step of the algorithm takes $O(NML)$ (for NM updates of Q-values, on N states and M actions, with each taking time L where L is the max number of states reachable by any action a from any state s.) **In general, however, policy iteration is considered the better algorithm in practice.**

One way to make policy iteration more efficient than value iteration is to dispense with solving Equation 5 exactly, and to be satisfied with an approximate solution. Note that Equation 5 is itself a fixed point equation, and can be solved by the same kind of iterative method we have been using for solving the optimality equations! We can start with an arbitrary value function, and apply the right hand side of Equation 5 for a fixed number of steps or until approximate convergence, to get better and better approximations to the V^{π}. This method of approximating the value function is called *value propagation*, and can be used instead of solving the equations in the policy iteration algorithm. We obtain the *modified policy iteration* algorithm, which tends to be more effective than value iteration and regular policy iteration.

Changelog

- 23 November 2018 – Initial version converted from Harvard CS181 course notes.