5
Policy Iteration

We saw in previous notes that value iteration can be used to find an optimal policy efficiently. Interestingly, in later rounds of value iteration, the best action at each state rarely changes. Put in other words, the policy implicitly defined by the value function appears to converge more rapidly than the value function itself. This insight suggests approaches that attempt to update an explicit estimate of the optimal policy rather than only an explicit estimate of optimal value function (with the policy implicit).

Policy Evaluation

In order to update the policy, we need some way to measure its performance. Fortunately, we have a way to do this: we can simply compute the value function for a fixed policy. We can use the value function $V^\pi(x,t)$ to denote the expected cost-to-go of a policy π in state x at time t. The process of finding V^π is called *policy evaluation*. We can use a policy evaluation algorithm to tell us how good one policy is to compare to others or suggest modifications.

Recall from the first chapter that we can compute the value function for a given policy π through the following algorithm:

Algorithm 8: Dynamic Program for creating an optimal value function on the infinite horizon by finite horizon approximation

Algorithm `EvaluatePolicy(`x, π,T`)`
  for $t = T-1,\ldots,0$ **do**
    for $x \in \mathbb{X}$ **do**
      if $t = T-1$ **then**
        $V(x,t) = c(x,\pi(x,t))$
      end
      else
        $V(x,t) = c(x,\pi(x,t)) + \gamma \sum_{x' \in \mathbb{X}} p(x'|x,\pi(x,t))V(x,t)$
      end
    end
  end
  return V

If π is stationary (not a function of time) [1] then as $t \to \infty$ the value function converges to fixed point satisfying the following *Bellman Equation*:

$$V^\pi(x,t) \xrightarrow{t\to\infty} V^\pi(x) = c(x,\pi(x)) + \gamma \sum_{x'} p(x'|x,\pi(x))V^\pi(x')$$

[1] Notably under assumptions that ensure convergence of the value function. If $\gamma < 1$ or, if, with probability 1, π enters a terminal state having zero cost.

Note that this equation is *linear* in $V^\pi(x)$. While this can be solved via policy iteration, an *alternate* way to compute this is to solve a system of linear equations.

Let $\overrightarrow{c^\pi}$ and $\overrightarrow{V^\pi}$ be vectors of length $|\mathbb{X}|$ listing the cost and cost-to-go, respectively for $\forall x \in \mathbb{X}$.

$$\overrightarrow{V^\pi} = \overrightarrow{c} + \gamma P^\pi \overrightarrow{V^\pi} \tag{5.0.1}$$

$$\Rightarrow (I - \gamma P^\pi)\, \overrightarrow{V^\pi} = \overrightarrow{c} \tag{5.0.2}$$

where P^π is the row stochastic transition matrix (its rows sum to 1) given the the fixed policy π

$$P^\pi = \begin{pmatrix} p(x_0|x_0,\pi(x_0)) & p(x_1|x_0,\pi(x_0)) & \dots \\ \vdots & \vdots & \vdots \\ p(x_0|x_n,\pi(x_n)) & p(x_1|x_n,\pi(x_n)) & \dots \end{pmatrix} \tag{5.0.3}$$

The operation of multiplying by P^π is the equivalent of calculating expectation. This is a linear equation in $\overrightarrow{V^\pi}$ and its solution is

$$\overrightarrow{V^\pi} = (I - \gamma P^\pi)^{-1} \overrightarrow{c} \tag{5.0.4}$$

For $\gamma < 1$ this equation always has a solution (the eigenvalues of P^π have modulus always less than one, so $I - \gamma P^\pi$ is always invertible).

Policy Improvement

If someone hands you a policy π, it is natural to want to see if it is optimal, and if not, to improve it (see Figure 5.0.1). Then, the question becomes, how can we tell that whether the policy is optimal? And, more importantly, if the policy is not optimal, "how can we modify the policy so that it becomes better, and eventually, optimal?" (See Figure 5.0.1) *Policy improvement* seeks to answer these questions.

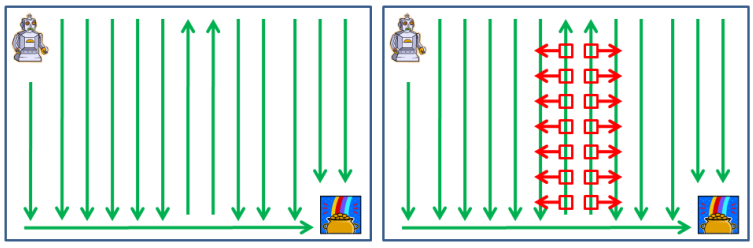


Figure 5.0.1: The left image shows a non-optimal policy (green arrows). The right image shows how the policy could be improved by changing the action taken on a state-by-state basis. The new policy is still not optimal and could be improved by another round of policy iteration.

In the *policy improvement* step, the policy is modified as follows $\forall x \in \mathbb{X}$:

$$\pi'(x) = \underset{a}{\operatorname{argmin}}\, c(x,a) + \gamma\, \mathbb{E}_{p(x'|x,a)}[V^\pi(x')]. \tag{5.0.5}$$

We can show that the new policy π' given by (5.0.5) will be at least as good as π. Moreover, as we shall see later, if the resulting policy π' is the same as the current policy π, then π is the already the optimal policy. Put differently, we only have an optimal policy if there is no change at any single state we can make that would appear to incur less long term cost.

Policy improvement can also be expressed in terms of $Q^\pi(x,a)$, the *quality function*, sometimes called the *Q-function* or *action value function*. The Q-function $Q^\pi(x,a)$ is the sum of the cost of performing an action a at state x and the expected cost to go from the resulting state under policy π.

$$Q^\pi(x,a) = c(x,a) + \gamma \, \mathbb{E}_{p(x'|x,a)}[V^\pi(x')] \tag{5.0.6}$$

A new policy π' can, therefore, be formed from an existing policy π by tweaking the action selected at a state. According to the policy improvement step, if π' is selected such that

$$\pi'(x) = \operatorname*{argmin}_a Q^\pi(x,a). \tag{5.0.7}$$

Policy Iteration Algorithm

Combining policy evaluation and policy improvement, we can get an algorithm, *Policy Iteration*, for finding a good policy from an arbitrary initial policy π_0.

Algorithm 9: Policy Iteration. Here the policy evaluation step can be computed by, e.g. Algorithm 8 or solving a linear system.

Start with arbitrary π_0
$k \leftarrow 0$
while *not converged* **do**
  Policy Evaluation: compute V^{π_k}
  for $\forall x \in \mathbb{X}$ **do**
    $\pi_{k+1}(x) = \operatorname*{argmin}_a c(x,a) + \gamma \sum_{x' \in \mathbb{X}} p(x'|x,a) \, V^{\pi_k}(x')$
  $k \leftarrow k+1$
return $\pi_k(x), \forall x$

5.1 *Policy Iteration Optimality*

During the policy iteration, the difference in value of the current policy π and the optimal value function $|V^\pi(x) - V^*(x)|$, decreases *exponentially* as a function of number of iterations. In practice– although with very little theoretical justification– it is found that policy iteration generally requires fewer iterations than Value Iteration. However, it does require more work on each iteration.

Understanding whether Policy Iteration will converge to the best policy is not trivial. The standard argument, outlined below, uses contradiction to show that there are no local optima, so, since each step is an improvement, the algorithm will converge to the optimum. To see this we need to show that:

- Policy Iteration monotonically improves
- Policy Iteration only produces no change in policy if it is at a global optima.

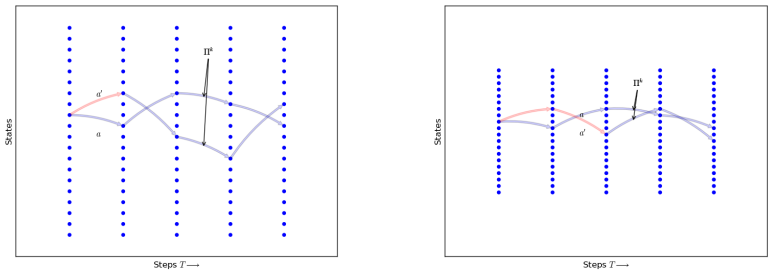
Together these imply reaching a global optima in finite time if there are a finite number of policies being considered. [2]

[2] Interestingly, it can be shown PI can theoretically visit an exponentially large set of policies, however, the policies distance from optimality decays geometrically in the number of iterations.

Monotonic Improvement

To show that the algorithm monotonically improves, we look at the improvement in the value function between policies. We switch actions *only if* (see Figure 5.1.1) the policy from that point onwards is an improvement.

To calculate the policy is improved, let us consider the difference between the value functions under two policies π and π' for a given initial state x_0. As is shown in Figure 5.1.1, let us consider the "value improvement" time-step by time-step.



(a) Choosing action a' in state x_0 which minimizes (5.0.5) at x_0, then following π

(b) Choosing action a' in state x_1 which minimizes (5.0.5), then following π

Figure 5.1.1: Improvement in the value function: Blue dots denote states, red arrows denote actions that minimizes (5.0.5) for a state, blue arrows denote actions in π.

The value improvement accrued at the initial time-step, due to the fact that π' chooses the action that minimizes (5.0.5) at x_0, must be:

$$\begin{aligned} & c(x_0, \pi'(x_0)) + \gamma \, \mathbb{E}_{p(x'|x_0,\pi'(x_0))}[V^\pi(x')] - V^\pi(x_0) \\ =& Q^\pi(x_0, \pi'(x_0)) - V^\pi(x_0). \end{aligned}$$

The value improvement accrued at the second time step (due to the fact that π' chooses the action that minimizes (5.0.5) at x_1) is:

$$\begin{aligned} & \mathbb{E}_{x_1 \sim p_1}\left[c(x_1, \pi'(x_1)) + \gamma \, \mathbb{E}_{p(x'|x_1,\pi'(x_1))}[V^\pi(x')] - V^\pi(x_1)\right] \\ =& \mathbb{E}_{x_1 \sim p_1}\left[Q^\pi(x_1, \pi'(x_1)) - V^\pi(x_1)\right], \end{aligned}$$

where $p_1(x_1) = p(x_1|x_0, \pi'(x_0))$.

Let's denote by $p_t(x)$ the probability of visiting state x at time t when we start at state x_0 and follow policy π', i.e. $p_t = Pr[x_t = x | x_o, \pi']$. By proceeding inductively from the above, we see that the difference between value functions can be calculated using the following equality:

Lemma 1. *Performance Difference Lemma:* [3]

$$V^{\pi'}(x_0) - V^\pi(x_0) = \sum_{t=0}^{\infty} \gamma^t \, \mathbb{E}_{x \sim p_t}\left[Q^\pi(x, \pi'(x)) - V^\pi(x)\right]$$

By the policy improvement step, we know that $Q^\pi(x, \pi'(x)) - V^\pi(x) \leq 0$ for all $x \in \mathbb{X}$. This lemma implies $V^{\pi'}(x_0) - V^\pi(x_0) \leq 0$ holds uniformly over all initial states and thus we see that the policy iteration algorithm improves the policy monotonically. [4]

[3] J. A. Bagnell, A. Y. Ng, S. Kakade, and J. Schneider. Policy search by dynamic programming. In *Advances in Neural Information Processing Systems*, 2003

[4] Assuming everywhere the infinite horizon value function exists, which of course holds for $\gamma < 1$

The Performance Difference Lemma is a powerful tool. It's proof, and the proof that policy improvement works in the Policy Iteration algorithm, have the same essential character.

Optimality

When policy iteration has stopped making improvements, i.e. a local optimum is reached,

$$(\pi, V^{\pi}) = (\pi', V^{\pi'}).$$

In this case, we have,

$$V^{\pi}(x) = V^{\pi'}(x) = \min_{a} c(x,a) + \gamma \, \mathbb{E}_{p(x'|x,a)} \left[V^{\pi}(x') \right].$$

Note this immediately implies that (π, V^{π}) are a solution to the Bellman Equation. Therefore $\pi' = \pi^*$ since (π^*, V^*) since the optimal value function– the Bellman Equation– is unique.

5.2 *Implementation Notes*

Often a *Modified Policy Iteration* is used in practice. Modified Policy Iteration warm-starts the policy evaluation step with the value function from the previous step and then does a few iterations k of policy evaluation. In the special case of $k = 1$, it reduces to VI (see Sutton and Barto Chapter 4). Since the expensive part of policy iteration is the policy evaluation step, this warm-start can greatly speed up the algorithm.

Dynamic programming algorithms (Value Iteration, Policy Iteration, Modified Policy Iteration, etc) are expensive if the state space is large. It can be used in its closed form (solving a linear system) if the value function is sparse. Otherwise the value function can be approximated by:

- Linear function approximator $\tilde{V}_{\theta}(x) = \theta^{\top} \phi(x)$, where $\phi(x)$ is the *feature* of the state x.
- Nearest Neighbour – for any x find the closest x' (in the sampled space) and return that value.
- Neural Network – a popular choice that has led to state of the art results in games from Backgammon to Chess.
- Any other regression algorithm....

Approximating the value function is the basis for the *Fitted Value Iteration* algorithm which we discuss in later lectures.

5.3 *Related Reading*

[1] Bagnell, J. A. , Kakade, S. Ng, A., Schneider, J. Policy Search by Dynamic Programming, NIPS, 2003.

[2] Puterman, M. Markov Decision Processes: Discrete Stochastic Dynamic Programming, 2005.