IEOR 265 - Lecture 10
Policy Iteration

1 Rollout Algorithm

We start the analysis by revisiting the Rollout algorithm for finite-horizon DP, where we solve a 1-step lookahead minimization problem:

$$\tilde{\mu}_k(x_k) \in \arg\min_{u_k \in U_k(x_k)} \left\{ \mathbb{E}_{w_k} \left[g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}(f_k(x_k, u_k, w_k)) \right] \right\}, \forall k \in \{0, ..., N-1\} \tag{1}$$

where the cost-to-go approximation $\tilde{J}_{k+1}(x_{k+1})$ is the total cost of some *Base Policy* $\hat{\pi} = (\hat{\mu}_{k+1}, ..., \hat{\mu}_{N-1})$, where the states are obtained by simulating the dynamics forwards for a sequence of sampled disturbances $(w_k, ..., w_{N-1})$:

$$x_{i+1} = f_i(x_i, \hat{\mu}_i(x_i), w_i), \ \forall i = k+1, ..., N-1 \tag{2}$$

Then we can leverage Sample-Average Approximation (SAA) to compute the approximate Q-factors:

$$\tilde{Q}_k(x_k, u_k) = \mathbb{E}_{w_k}[g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}(f_k(x_k, \hat{\mu}_k(x_k), w_k))] \approx$$
$$\sum_{s=1}^{S} r_s \big(g_k(x_k, u_k, w_k^s) + \tilde{J}_{k+1}(f_k(x_k, \hat{\mu}_k(x_k), w_k^s)) \big) \tag{3}$$

where S is the total number of samples and r_s is the "weight" of each scenario(for example $r_s = 1/S$, $s \in \{1, ..., S\}$). Then, the **rollout policy** becomes:

$$\tilde{\mu}_k(x_k) \in \min_{u_k \in U_k(x_k)} \Big\{ \sum_{s=1}^{S} r_s \big(g_k(x_k, u_k, w_k^s) + \tilde{J}_{k+1}(f_k(x_k, \hat{\mu}_k(x_k), w_k^s)) \big) \Big\} \tag{4}$$

We note the interesting feature of the rollout algorithm: we start with a base policy $\hat{\mu}_k$; then it generate a new rollout policy $\tilde{\mu}_k$. We showed in a previous lecture that under some conditions, the rollout policy represents an improvement over the base policy. We can then use the rollout policy as a new base policy and repeat the algorithm again. This repeated application leads to a *perpetual* Rollout algorithm that sequentially generates a sequence of policies:

$$\tilde{\mu}_k^{(0)}(x_k) \to \tilde{\mu}_k^{(1)}(x_k) \to \tilde{\mu}_k^{(2)}(x_k) \to \cdots \tag{5}$$

In finite-horizon DP, the perpetual rollout algorithm can achieve policy improvement for the entire sequence of policies. Now, as we did for the Value Iteration, we will extend the perpetual rollout algorithm to the infinite-horizon setting, which will leads us to the **Policy Iteration** algorithm.

2 Infinite-Horizon DP and Policy Iteration

As before, it suits our purposes to work with the Markov-Decision-Process (MDP) formulation. To that end, we use i to denote the states, from a set of integers $i \in \{1, ..., n\}$; we let $U(i)$ be the set of available controls at state i; we define $p_{ij}(u)$ as the probability of transitioning from i to j, given that the chosen control is u; and lastly we define the stage cost to be $g(i, u, j)$ on said transition. Our goal, in the infinite-horizon setting, is still to solve the **Bellman's Equation**:

$$J^*(i) = \min_{u \in U(i)} \left\{ \sum_{j=1}^{n} p_{ij}(u)(g(i, u, j) + \alpha J^*(j)) \right\} \tag{6}$$

On VI, we solved it upon convergence: We sequentially generate a sequence of cost-to-go functions $\{J^{(t)}(\cdot)\}_{t \geq 0}$, which converge to the solution of the Bellman's Equation. Then the optimal *stationary* policy is read directly from the right-hand side of Eq(6). On Policy Iteration (PI), we will generate a sequence of stationary policies, and hope that they will converge to a policy that solves the Bellman's Equation. Intuitively, we can note that in PI we go in the "opposite way" of VI. Nevertheless both of them seek to solve the same problem, namely Eq(6).

The PI algorithm begins with a stationary base policy $\mu^{(t)}$, where we use the superscript t to denote the algorithm iterations. Then, it computes the cost-to-go values $J_{\mu^{(t)}}(1), ..., J_{\mu^{(t)}}(n)$ associated with $\mu^{(t)}$, by solving the following system of *linear* equations:

$$J_{\mu^{(t)}}(i) = \sum_{j=1}^{n} p_{ij}(\mu^{(t)}(i)) \big(g(i, \mu^{(t)}(i), j) + \alpha J_{\mu^{(t)}}(j) \big), \; \forall i \in \{1, ..., n\} \tag{7}$$

This is called the **policy evaluation step**: we evaluate the policy $\mu^{(t)}$ by solving a "version" of the Bellman's Equation where we stick to the policy $\mu^{(t)}$, instead of solving any minimization. Next, the algorithm uses the computed cost-to-go values $J_{\mu^{(t)}}(1), ..., J_{\mu^{(t)}}(n)$ to compute a new policy $\mu^{(t+1)}$ as:

$$\mu^{(t+1)}(i) \in \arg\min_{u \in U(i)} \left\{ \sum_{j=1}^{n} p_{ij}(u) \big(g(i, u, j) + \alpha J_{\mu^{(t)}}(j) \big) \right\}, \; \forall i \in \{1, ..., n\} \tag{8}$$

This is the **policy improvement step**: note that Eq(8) is similar to a 1-step lookahead minimization where the cost-to-go approximation is given by the cost-to-go values associated with the old policy $\mu^{(t)}$. So, in essence, the policy improvement step is the Rollout algorithm, where $\mu^{(t)}$ plays the role of the base policy and $\mu^{(t+1)}$ plays the role of the rollout policy. The PI algorithm alternates between these two steps sequentially, until:

$$J_{\mu^{(t+1)}}(i) = J_{\mu^{(t)}}(i), \; \forall i \in \{1, ..., n\} \tag{9}$$

A natural question to follow is whether such scheme converge to anything, and if it does converge, whether the policy obtained actually solves the Bellman's Equation. It turns out that the PI algorithm converges to the optimal stationary policy, which solves the Bellman Equation, Eq(6). This is formally state in the following theorem 1.

Theorem 1 ***(Converge of PI):*** *Given any initial stationary policy $\mu^{(0)}$, the sequence $\{\mu^{(0)}\}_{t\geq 0}$ generated by the PI Algorithm, have the policy improvement property:*

$$J_{\mu^{(t+1)}}(i) \leq J_{\mu^{(t)}}(i),\ \forall i \in \{1,...,n\}\ \textit{and}\ t \geq 0 \tag{10}$$

And the algorithm terminates with the optimal stationary policy that solves the Bellman Equation:

$$J^*(i) = \min_{u\in U(i)} \left\{ \sum_{j=1}^{n} p_{ij}(u)(g(i,u,j) + \alpha J^*(j)) \right\} \tag{11}$$

proof: The proof of this result is fairly simple and has a nice intuition that ties with the *policy improvement* property we saw in the Rollout algorithm, for finite-horizon DP. It follows closely the development in [1]. Let μ be our starting policy and $\bar{\mu}$ be the policy generated by one iteration of the PI algorithm. Our goal is to show that:

$$J_{\bar{\mu}}(i) \leq J_{\mu}(i),\ \forall i \in \{1,...,n\} \tag{12}$$

Let J_N denote the cost of a policy that applies $\bar{\mu}$ for the first N stages, and then applies μ for every subsequent stages. We first state the Bellman's Equation associated with the policy μ:

$$J_{\mu}(i) = \sum_{j=1}^{n} p_{ij}(\mu(i))\big(g(i,\mu(i),j) + \alpha J_{\mu}(j)\big) \tag{13}$$

Now, by the policy improvement step, it follows that:

$$\bar{\mu}(i) \in \arg\min_{u\in U(i)} \left\{ \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J_{\mu}(j)\big) \right\},\ \forall i \in \{1,...,n\} \tag{14}$$

which gives us J_1:

$$J_1(i) = \sum_{j=1}^{n} p_{ij}(\bar{\mu}(i))\big(g(i,\bar{\mu}(i),j) + \alpha J_{\mu}(j)\big) \leq J_{\mu}(i)$$

Reiterating, J_1 is the cost of a policy that applies $\bar{\mu}$ for the first stage, and then applies μ for every subsequent stages. Now if we replace $J_{\mu}(\cdot)$ by J_1 we obtain J_2:

$$J_2(i) = \sum_{j=1}^{n} p_{ij}(\bar{\mu}(i))\big(g(i,\bar{\mu}(i),j) + \alpha J_1(j)\big) \leq J_1(i) \tag{15}$$

Then we have the inequality:

$$J_2(i) \leq J_1(i) \leq J_{\mu}(i),\ \forall i \in \{1,...,n\} \tag{16}$$

Now, proceeding inductively, we conclude that:

$$J_{N+1}(i) \leq J_N(i) \leq J_{\mu}(i),\ \forall i \in \{1,...,n\} \text{ and } \forall N \geq 1 \tag{17}$$

Now taking the limit of $N \to \infty$, it follows that $J_N(i) \to J_{\bar{\mu}}(i)$ for all $i \in \{1, ..., n\}$ and we have:

$$J_{\bar{\mu}}(i) \leq J_{\mu}(i), \forall i \in \{1, ..., n\} \tag{18}$$

So we have proved the policy the improvement policy. Now, since the number of stationary policies is finite, the algorithm must terminate, after a finite number of iterations, say t, where it holds that:

$$J_{\mu^{(t+1)}}(i) = J_{\mu^{(t)}}(i), \forall i \in \{1, ..., n\} \tag{19}$$

Then it means equality will hold throughout Eq(17) and:

$$J_{\mu^{(t)}}(i) = \min_{u \in U(i)} \left\{ \sum_{j=1}^{n} p_{ij}(u)(g(i,u,j) + \alpha J_{\mu^{(t)}}(j)) \right\}, \forall i \in \{1, ..., n\} \tag{20}$$

Hence the costs $J_{\mu^{(t)}}(1), ..., J_{\mu^{(t)}}(n)$ solve the Bellman's Equation, therefore $\mu^{(t)}(i)$ is the optimal stationary policy. **Q.E.D.**

2.1 Example: Treasure Hunting

We will illustrate the PI algorithm with a treasure hunting problem. Consider a profit-maximizing agent that every day can decide to explore a site with a total of n treasures or go home. If the agent goes home, they can never return to the site. If they decide to explore/investigate the site, they incur a fixed cost of c and may find any amount of treasures with some positive probability. Assume that every treasure is worth 1 and that if there are i treasures left on the site and the agent decides to explore than they find a total of $m \in [0, i]$ treasures with probability $p(m|i)$. And we further assume that $p(0|i) < 1$, for all $i \geq 1$. We can represent this problem in the following schematic figure:

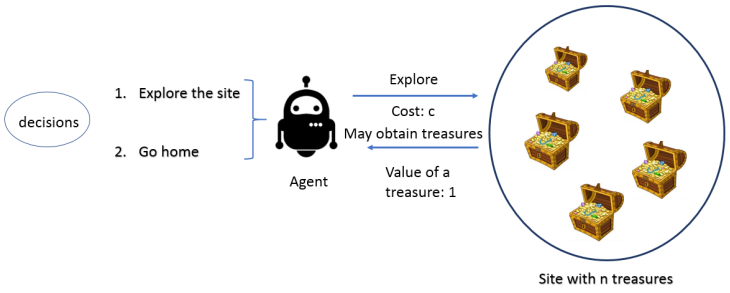

Figure 1: Schematic figure of the treasure hunting problem: An agent needs to decide every day to either explore the site with treasures or go home. Upon going home, they may never return, but exploring incurs a fixed cost with an uncertain reward.

Hence, if there are i treasures remaining in the site, the expected reward to be found is:

$$r(i) = \sum_{m=0}^{i} mp(m|i) \tag{21}$$

and we assume that $r(i)$ is monotonically increasing with i (so the more treasures on the site, the more the agent expects to make by exploring it). For this problem we let the states of the MDP be the amount of treasures **not yet** found plus an additional termination state 0, that indicate that either the agent decided to go home or there are no more treasures to be found. The controls are either "explore" or "go home".

We start by observing that if we are on state $i \geq 1$, then we either move to state 0 with probability 1, if we decide to go home; or we move to state $i - m$ with probability $p(m|i)$, if we decide to explore. Then we can write the Bellman's Equation as:

$$J^*(i) = \max\left\{0, r(i) - c + \sum_{m=0}^{i} p(m|i)J^*(i-m)\right\},\ i \in \{1, ..., n\} \tag{22}$$

and $J^*(0) = 0$. Let's start the PI algorithm using as the first policy $\mu^{(0)}$ that *never* explores, that is it always elects to go home regardless of which state. Performing the policy evaluation step yields:

$$J_{\mu^{(t)}}(i) = \sum_{j=1}^{n} p_{ij}(\mu^{(t)}(i))\big(g(i, \mu^{(t)}(i), j) + \alpha J_{\mu^{(t)}}(j)\big) \tag{23}$$

which, in this case, reduces to:

$$J_{\mu^{(0)}}(i) = 0,\ i \in \{1, ..., n\} \tag{24}$$

Now, we perform the policy improvement step:

$$\mu^{(t+1)}(i) \in \arg\min_{u \in U(i)} \left\{ \sum_{j=1}^{n} p_{ij}(u)\big(g(i, u, j) + \alpha J_{\mu^{(t)}}(j)\big) \right\},\ \forall i \in \{1, ..., n\} \tag{25}$$

which, in this case, reduces to:

$$\mu^{(1)}(i) \in \arg\max\big\{0, r(i) - c\big\},\ \forall i \in \{1, ..., n\}$$

this policy can be written equivalently as:

$$\mu^{(1)}(i) = \begin{cases} \text{"explore"}, \text{ if } r(i) > c \\ \text{"go home"}, \text{ otherwise} \end{cases} \tag{26}$$

which we can state in words: "we explore if the expected reward to be found in the site is larger than the fixed cost; otherwise we go home". Now we perform the policy evaluation step again, yielding:

$$J_{\mu^{(1)}}(i) = \begin{cases} 0, & \text{if } r(i) \leq c \\ r(i) - c + \sum_{m=0}^{i} p(m|i)J_{\mu^{(1)}}(i-m), & \text{if } r(i) > c \end{cases} \tag{27}$$

and we can verify the policy improvement property. Indeed, we have that:

$$J_{\mu^{(1)}}(i) \geq J_{\mu^{(0)}}(i) = 0,\ \forall i \in \{1, ..., n\}$$

where we note that we are trying to maximize profit (so by "improvement" we mean larger profits). Now, we perform the policy improvement step once more:

$$\mu^{(2)}(i) \in \arg\max\big\{0, r(i) - c + \sum_{m=0}^{i} p(m|i)J_{\mu^{(1)}}(i-m)\big\},\ \forall i \in \{1, ..., n\} \tag{28}$$

Now observe if $r(i) \leq c$ then it follows that $r(j) \leq c$ for all $j < i$, since $r(i)$ is monotonically increasing. Then it follows that for all i such that $r(i) \leq c$:

$$0 \geq r(i) - c + \sum_{m=0}^{i} p(m|i) J_{\mu^{(1)}}(i-m) \tag{29}$$

so we have that:

$$\mu^{(2)}(i) = \text{ “go home ”, } \forall i \text{ s.t.: } r(i) \leq c \tag{30}$$

Now if $r(i) > c$, since:

$$J_{\mu^{(1)}}(i) \geq J_{\mu^{(0)}}(i) = 0, \, \forall i \in \{1, ..., n\} \tag{31}$$

it follows that:

$$0 < r(i) - c + \sum_{m=0}^{i} p(m|i) J_{\mu^{(1)}}(i-m) \tag{32}$$

so we have that:

$$\mu^{(2)}(i) = \text{ “explore ”, } \forall i \text{ such that } r(i) > c \tag{33}$$

Patching it together, we can state the new policy as:

$$\mu^{(2)}(i) = \begin{cases} \text{“explore”, if } r(i) > c \\ \text{“go home”, otherwise} \end{cases}$$

And we note that $\mu^{(2)}(i) = \mu^{(1)}(i)$ for all $i \in \{0, ..., n\}$. So the PI algorithm converged and we are done! That is the optimal policy and it solves the Bellman's Equation Eq(22)

2.2 Q-factors and Optimistic PI algorithm

On this subsection, we provide two variations of PI: one that uses Q-factors; and one that uses the VI algorithm in place of the policy evaluation step. First note that we can write both the policy evaluation and policy improvement steps, in terms of the Q-factors:

$$\begin{aligned} Q_\mu(i,u) &= \sum_{j=1}^{n} p_{ij}(u)(g(i,u,j) + \alpha J_\mu(j)), \; i \in \{1, .., n\}, u \in U(i) \\ J_\mu(j) &= Q_\mu(i, \mu(j)) \end{aligned} \tag{34}$$

where $Q_\mu(i,u)$ is the Q-factor of the state-action pair (i,u) associated with the policy μ. The policy evaluation step becomes:

$$Q_{\mu^{(t)}}(i,u) = \sum_{j=1}^{n} p_{ij}(\mu^{(t)}(i)) \big(g(i,u,j) + \alpha Q_{\mu^{(t)}}(j, \mu^{(t)}(j))\big), \; \forall i \in \{1, ..., n\}, u \in U(i) \tag{35}$$

and the policy improvement step becomes:

$$\mu^{(t+1)}(i) \in \arg\min_{u \in U(i)} \Big\{ Q_{\mu^{(t)}}(i,u) \Big\}, \; \forall i \in \{1, ..., n\} \tag{36}$$

Note the typical trade-off found in using Q-factors is also present here: the policy improvement is much easier to compute since we do not require knowledge of the underlying MDP (as the transition probabilities are not used); however, the policy evaluation step becomes more expensive since now the linear system involves many more variables, namely all state-control pairs, instead of just the number of states.

We finish this lecture by addressing the very fact that the policy evaluation step requires solving a system of linear equations. Even though there are efficient ways of solving linear equations, for example by gaussian elimination, the linear system:

$$J_{\mu^{(t)}}(i) = \sum_{j=1}^{n} p_{ij}(\mu^{(t)}(i))\big(g(i, \mu^{(t)}(i), j) + \alpha J_{\mu^{(t)}}(j)\big)$$

on the variables $J_{\mu^{(t)}}(1), ..., J_{\mu^{(t)}}(n)$ may be very costly to solve for very large n. We can then resort to "partially" solve the linear system, by instead of computing the exact solution we apply some iterative scheme for a few iterations (an intuition here is to draw a parallel with Newton's method in root-finding problems). The iterative scheme we will utilize is the VI algorithm. This combination of PI and VI, where the policy evaluation step is performed by the VI algorithm is called **Optimistic** or **Generalized** PI algorithm. It starts with some cost-to-go function $J^{(t)}$ and performs the policy improvement step exactly as before:

$$\mu^{(t)}(i) \in \arg\min_{u \in U(i)} \left\{ \sum_{j=1}^{n} p_{ij}(u)\big(g(i, u, j) + \alpha J^{(t)}(j)\big) \right\}, \forall i \in \{1, ..., n\} \qquad (37)$$

in order to obtain a new policy $\mu^{(t)}$. Then, starting with $\hat{J}_0^{(t)} = J^{(t)}$, we apply m_t VI-steps for policy $\mu^{(t)}$ to compute $\hat{J}_1^{(t)}, ..., \hat{J}_{m_t}^{(t)}$ according to:

$$\hat{J}_{m+1}^{(t)}(i) = \sum_{j=1}^{n} p_{ij}(\mu^{(t)}(i))\big(g(i, \mu^{(t)}(i), j) + \alpha \hat{J}_m^{(t)}\big) \Big\}, \forall i \in \{1, ..., n\} \qquad (38)$$

for all $m \in \{0, ..., m_t - 1\}$ and sets $J^{(t+1)} = \hat{J}_{m_t}^{(t)}$. The optimistic PI algorithm then iterates between both steps Eq(37) and Eq(38). It turns out that optimistic PI algorithm also converges to the solution of the Bellman's Equation, but it's analysis a bit more involved and we shall not cover it here.

The optimistic PI algorithm is interesting because it is a "step" in the direction of solving the computations issues involving exact PI. Here, by "exact" we mean that we have perfect knowledge of the underlying MDP and we can perform the policy improvement and policy evaluations exactly.

As we did in VI, on the next lecture we will introduce approximation architectures, such as Deep Neural Networks, in the PI framework. Lastly, we observe that the optimistic PI algorithm can be further improved by expanding the policy improvement step as a multi-step lookahead minimization (instead of 1-step) as well, while still maintaining the convergence guarantees. Overall, the optimistic PI framework is flexible and it lies as the foundation stone from where to introduce approximations.

References

[1] D. P. Bertsekas, *Reinforcement learning and optimal control.* Athena Scientific, 2019.