Chapter 05 – Approximate Policy Iteration

Author: Gianmarco Scarano

gianmarcoscarano@gmail.com

1. State Visitation Probability

So, we know that Policy Iteration only makes progress and the performance progress of the policy is monotonic. What's the probability of visiting state s, a at time t according to a policy π starting from s_0?

$$d_{s_0}^{\pi}(s,a) = (1-\gamma)\sum_{h=0}^{\infty}\gamma^h\,\mathbb{P}_h^{\pi}(s,a;\ s_0)$$

Well, we know that $\mathbb{P}_t^{\pi}(s,a;\ s_0) = \sum_{a_0,s_1,a_1,\dots,s_{t-1},a_{t-1}}\mathbb{P}^{\pi}(s_0,a_0,..,s_t=s,a_t=a)$.

The problem arises when we deal with a state-space that is really large and we can't perform exact iterative policy evaluation for all the states (think about Tetris, Go or continuous spaces).

2. Approximate Policy Iteration

Assuming that the Infinite-Horizon MDP is still known, the state-space is too large to just enumerate all states and compute $V^{\pi}(s)$.

$MDP = (S, A, R, T, \gamma, \mu_0)$ where μ_0: Initial state distribution and Q is in $[0, \frac{1}{1-\gamma}]$.

At every iteration, we are outputting policies: $\{\pi_0, \pi_1, \pi_2, \dots \pi_T\}$

STEPS:

1. Start with a random guess π_0.
2. For $t = 0, \dots\ , T$:
    a. Do policy **evaluation** and compute $Q^{\hat{}\pi t}$ like this:
        i. $Q^{\hat{}\pi}(s_t, a) = r_t + \gamma\mathbb{E}_{s' - p(\cdot\,|\,s,a)}[V^{\hat{}\pi}(s')]$
    b. Do policy **improvement** as:
        i. $\pi_{t+1} = argmax_a Q^{\hat{}\pi t}(s,a)$ (We can still do argmax since we can still enumerate actions or discretize them).

Here, we are not computing Q for all the states and actions. We are just approximating Q and V as well. In this case if $V^{\hat{}\pi} \approx V^{\pi}$ then we can conclude that both are really close to each other, so the optimal policy will be close-to-optimal. What we are doing is that we are using a function approximator (Linear approximators, Neural Networks, etc.) in order to generalize among states, avoid looking at the whole S.

3. Data and Least Square Regression

Actually, we can directly approximate Q, and because of this we get rid of the assumption of knowing the MDP.

Declaring $D = \{s_i, a_t, y_i\}_{i=1}^N$ with y being our label, we can then use the Least-Square regression to extract a function Q in the family of functions: SxA $(State - Action) \rightarrow [0, \frac{1}{1-\gamma}]$

Such that we want to find the argmin of Q in this way:

$$argmin_Q \text{ in } \sum_{i=1}^{N}(Q(s_i, a_i) - y_i)^2$$

where y_i is our ground truth and $Q(s_i, a_i)$ is our computed value. This problem is a numerically tractable regression problem.

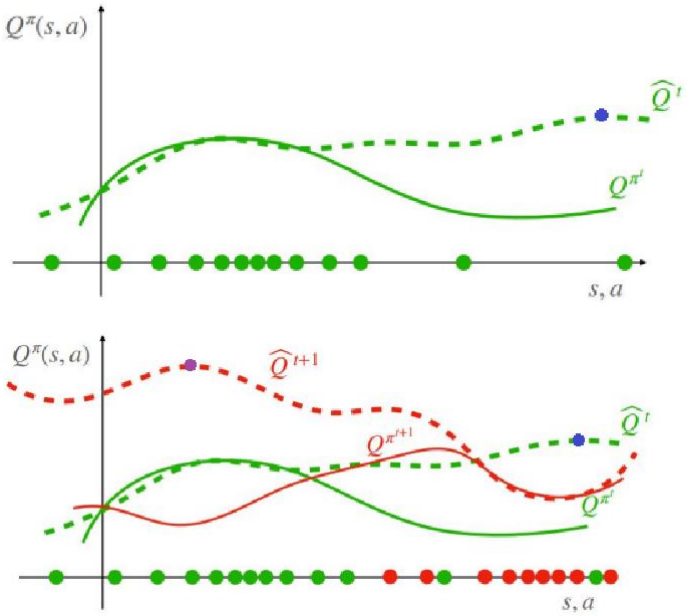
4. Oscillation from Distribution Change

In this case, when we compute our approximation of Q (Q^), we can't guarantee anymore monotonic improvement due to the fact that if we don't have enough data in the approximated Q^, we might end up in an extrapolation problem, explained below.

We define:

- **Green dots**: $(s, a)\ from\ \pi^t$
- **Red dots**: $(s, a)\ from\ \pi^{t+1}$

So, build our approximation knowing the ground truth Q^{π^t}. Our approximation is: $Q\hat{}^t$ (which is the green dotted line). Our argmax of the approximation tells us that we must rely on the space of the **blue** dot. In fact, at the next iteration, we don't have samples at the beginning because policy is focused on the (s, a) from previous iteration.



In fact, at next iteration $t + 1$, we see that (as stated before), we have samples in the region of the blue point, while the argmax of the approximated function (**purple** dot) says that we must return to the previous region.

This doesn't guarantee monotonic improvement. The problem comes directly from data because it comes from Q, while our estimation is only good under the same data distribution $d^{\pi}_{\mu_0}$. Lastly, to make sure we have monotonic improvement we need a strong coverage assumption.

5. Data Generation

2 main steps to consider:

- *Roll-in*
- *Roll-out* & compute supervision targets

$$D = \{S_i, Q_i, Y_i\}$$

5.1 Roll-in

In this case, in order to generate data, we want to sample our:

$$(s, a) \sim d^{\pi}_{s_0}(s, a) = (1 - \gamma) \sum_{h=0}^{\infty} \gamma^h\, \mathbb{P}^{\pi}_h(s, a;\ s_0)$$

Meaning that we can sample as much as possible (s, a) from the State Probability Distribution that we have in our dataset.

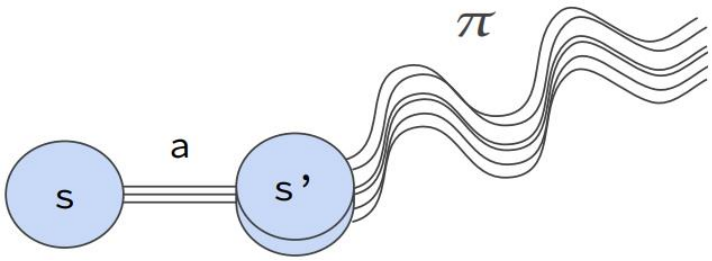
A little recall: When we have to sample from mixtures, we sample from mixture $p = (1 - \alpha)p_1 + \alpha p_2$, where we flip a coin with probability $[\alpha, 1 - \alpha]$. We commit to and sample from a specific p_i based on that.

Now, we want to sample h from $\frac{\gamma^h}{1-\gamma}$,thus committing to a specific $\mathbb{P}_h^{\pi}(s, a;\ s_0)$ $(Probability\ of\ a\ certain\ action - space)$ and follow π for h timesteps starting from $s_0 \sim \mu_0$, getting s_h, a_h.

In other words, we would like to use, given s_0, the probability of a certain action-space (mixture), according to h we sampled. The timesteps that we get are sampled from h as well.

5.2 Roll-out and computing supervision targets

Now the question is, how can we estimate $Q^{\pi}(s, a)$ given (s, a)? We can simply sample many times and average the results in order to get an estimate.



Well, while this could theoretically work, we have to take into consideration that the Q function is calculated in this way:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}[\sum_{h=0}^{\infty} \gamma^h r_h \mid (s_0, a_0) = (s_t, a_t), a_{h+1} = \pi(s_h), s_{h+1} \sim p(\cdot \mid s_h, a_h)]$$

Which is the expectation of the sum of the discounted rewards, given that at the first state I could take any action and only from state s_{h+1} I will follow my policy π.

We now note that on top of the summation symbol we have an infinite symbol, so we're in an infinite horizon problem. We could then think of using γ as a sampling factor again to choose a horizon, inducing the problem to be a finite-horizon one instead.

STEPS:

- Start at s, a
- Repeat:
  - Get reward of $(s, a) \Rightarrow r(s, a)$
  - With probability $1 - \gamma$ terminate and return $y = \sum \gamma^h r_h$
  - Execute the action and get in the next state s'.

In this way, we'll get our Data $D = \{s_i, a_i, y_i\}_{i=1}^{N}$