Reinforcement Learning

On- and Off-Policy Learning

Subramanian Ramamoorthy
School of Informatics

3 February, 2017

Can We Avoid Thorny Assumptions?

- Two major MC assumptions (infinite sampling and exploring all states) are unrealistic. How to circumvent the issue?

- Need to continually explore, ε-soft policies:

  - **On-policy** method: Explore in an ε-greedy manner

  - **Off-policy** method: Use a behaviour policy that is good at exploring, then infer optimal policy from that

On-Policy Monte Carlo Control

- Overall idea is still that of Generalized Policy Iteration (move *towards* greedy policy), but throw in continual exploration
- In order to always explore, we want to keep policy **ε-soft**:

$$\pi(s,a) > 0, \forall s, \forall a$$

- Moreover, one may really wish to adopt an **ε-greedy** policy:

$$\begin{aligned}\pi(s,a) &= \frac{\epsilon}{|\mathcal{A}|}, \text{if } a \textit{ is not} \text{ the greedy choice} \\ &= 1-\epsilon+\frac{\epsilon}{|\mathcal{A}|}, \text{if } a \textit{ is} \text{ the greedy choice}\end{aligned}$$

- In this case, we have $\pi(s,a) > \frac{\epsilon}{|\mathcal{A}|}, \forall s, \forall a$

On-Policy MC Control

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
    $Q(s,a) \leftarrow$ arbitrary
    $Returns(s,a) \leftarrow$ empty list
    $\pi \leftarrow$ an arbitrary ε-soft policy

Repeat forever:
    (a) Generate an episode using π
    (b) For each pair s, a appearing in the episode:
        $R \leftarrow$ return following the first occurrence of s, a
        Append R to $Returns(s,a)$
        $Q(s,a) \leftarrow \text{average}(Returns(s,a))$
    (c) For each s in the episode:
        $a^* \leftarrow \arg\max_a Q(s,a)$
        For all $a \in \mathcal{A}(s)$:

$$\pi(s,a) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(s)| & \text{if } a = a^* \\ \varepsilon/|\mathcal{A}(s)| & \text{if } a \neq a^* \end{cases}$$

Evaluate as before

Improve towards ε-greedy, not the max

The Policy Improvement Step

- Any ε-greedy policy w.r.t. Q^π is an improvement over any ε-soft policy π (Policy Improvement Theorem)

ε - greedy policy

$$
\begin{aligned}
Q^\pi(s,\pi'(s,a)) &= \sum_a \pi'(s,a) Q^\pi(s,a) \\
&= \frac{\epsilon}{|\mathcal{A}(s)|}\sum_a Q^\pi(s,a) + (1-\epsilon)\max_a Q^\pi(s,a) \\
&\geq \frac{\epsilon}{|\mathcal{A}(s)|}\sum_a Q^\pi(s,a) + (1-\epsilon)\sum_a \frac{\pi(s,a) - \frac{\epsilon}{|\mathcal{A}(s)|}}{1-\epsilon} Q^\pi(s,a) \\
& \text{This is bounded above by,} \\
&= \frac{\epsilon}{|\mathcal{A}(s)|}\sum_a Q^\pi(s,a) - \frac{\epsilon}{|\mathcal{A}(s)|}\sum_a Q^\pi(s,a) + \sum_a \pi(s,a) Q^\pi(s,a) \\
&= V^\pi(s)
\end{aligned}
$$

Off-policy Method

- Evaluate one policy while following another one
  - Behaviour policy takes you around the environment
  - Estimation policy is what you are after
- Of course, this requires: $\pi(s,a) > 0 \implies \pi'(s,a) > 0, \forall s, \forall a$
- Then, the off-policy procedure works as follows:
  - Compute the weighted average of returns from behaviour policy
  - Weighting factors are the probability of the moves being in estimation policy
  - i.e., weight each return by relative probability of being generated by π and π'

Learning a Policy while Following Another

On the ith first visit to state s, let:

$p'_i(s) =$ probability of getting subsequent sequence of states and actions from π' (BEHAVIOUR) $\quad T$ is the end-of-episode time

Using this to get data

$$p'_i(s_t) \quad = \quad \prod_{k=t}^{T_i(s)-1} \pi'(s_k, a_k) P^{a_k}_{s_k s_{k+1}}$$

$R'_i(s) =$ return observed from following the behaviour policy through this sequence of states and actions

Learning a Policy while Following Another

Let $p_i(s)$ = probability of getting the same sequence of states and actions from π (ESTIMATION)

$$p_i(s_t) = \prod_{k=t}^{T_i(s)-1} \pi(s_k, a_k) P^{a_k}_{s_k s_{k+1}}$$

Then after n_s returns experienced from state s (so episodes in which s occurs), weight each return by relative probability of occurring in π and π' and average:

$$V^\pi(s) = \frac{\sum_{i=1}^{n_s} \frac{p_i(s)}{p'_i(s)} R'_i(s)}{\sum_{i=1}^{n_s} \frac{p_i(s)}{p'_i(s)}}$$

Comparing the two Probabilities

$$
\begin{aligned}
p_i(s_t) &= \prod_{k=t}^{T_i(s)-1} \pi(s_k, a_k) P^{a_k}_{s_k s_{k+1}} \\
p'_i(s_t) &= \prod_{k=t}^{T_i(s)-1} \pi'(s_k, a_k) P^{a_k}_{s_k s_{k+1}} \\
\frac{p_i(s_t)}{p'_i(s_t)} &= \prod_{k=t}^{T_i(s)-1} \frac{\pi(s_k, a_k)}{\pi'(s_k, a_k)}
\end{aligned}
$$

So the weighting factors don't depend on environment, only on the two policies. How can we use this?

Off-Policy MC Algorithm

How to use this formula to get Q-values?

- Use *Behaviour Policy* π' to generate moves
  - must be soft so that all (s, a) continue to be explored

- Evaluate and improve *Estimation Policy* π
  - converges to optimal policy

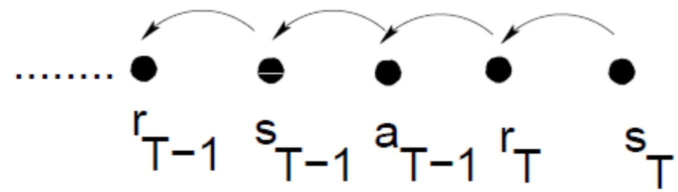
So...

1. BP π' generates episode

2. EP π is deterministic and gives the greedy actions w.r.t. the current estimate of Q^π (it is arbitrary for the first episode)

Off-Policy MC Algorithm, cont.

3. Start at end of episode, work backwards



till BP and EP give divergent actions, e.g. back to time t

4. For this chain of states and actions compute

$$\frac{p_i(s_t)}{p_i'(s_t)} = \prod_{k=t}^{T_i(s)-1} \frac{\pi(s_k, a_k)}{\pi'(s_k, a_k)}$$

π is deterministic so $\pi(s_k, a_k)$ etc. $= 1$ and we know π'

Off-Policy MC Algorithm, cont.

So

$$\frac{p_i(s_t)}{p_i'(s_t)} = \prod_{k=t}^{T_i(s)-1} \frac{1}{\pi'(s_k, a_k)}$$

5.

$$Q(s,a) = \frac{\sum \frac{p_i}{p_i'} R'}{\sum \frac{p_i}{p_i'}}$$

Sum is over no. times this (s, a) has been visited, say N

$R' =$ return for the chain of states/actions (see 3) following (s, a) (it's different for each of the N visits, as is p/p')

Off-Policy MC Algorithm, cont.

6. Do for each (s, a) in chain (see 3)

7. Improve π (estimation policy) to be greedy w.r.t. Q:

   $\pi(s) = \arg\max_a Q(s, a)$

   (Still deterministic, so still 1 for transitions within it.)

8. Back to 1. Repeat until estimation policy and Q values converge.

Takes a long time because we can only use the information from the end of the episode in each iteration.

The Off-Policy MC Control Algorithm

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:

$Q(s,a) \leftarrow$ arbitrary

$N(s,a) \leftarrow 0$; Numerator and

$D(s,a) \leftarrow 0$; Denominator of $Q(s,a)$

$\pi \leftarrow$ an arbitrary deterministic policy

Repeat forever:

(a) Select a policy π' and use it to generate an episode:

$$s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_{T-1}, a_{T-1}, r_T, s_T$$

(b) $\tau \leftarrow$ latest time at which $a_\tau \neq \pi(s_\tau)$

(c) For each pair s, a appearing in the episode after τ:

$t \leftarrow$ the time of first occurrence (after τ) of s, a

$w \leftarrow \prod_{k=t+1}^{T-1} \frac{1}{\pi'(s_k, a_k)}$

$N(s,a) \leftarrow N(s,a) + wR_t$

$D(s,a) \leftarrow D(s,a) + w$

$Q(s,a) \leftarrow \frac{N(s,a)}{D(s,a)}$

(d) For each $s \in \mathcal{S}$:

$\pi(s) \leftarrow \arg\max_a Q(s,a)$

Incremental Implementation

- Better to implement MC incrementally (think memory…)

- To compute the weighted average of each return:

$$V_n = \frac{\sum_{k=1}^{n} w_k R_k}{\sum_{k=1}^{n} w_k}$$

non-incremental

$$V_{n+1} = V_n + \frac{w_{n+1}}{W_{n+1}}\left[R_{n+1} - V_n\right]$$

$$W_{n+1} = W_n + w_{n+1}$$

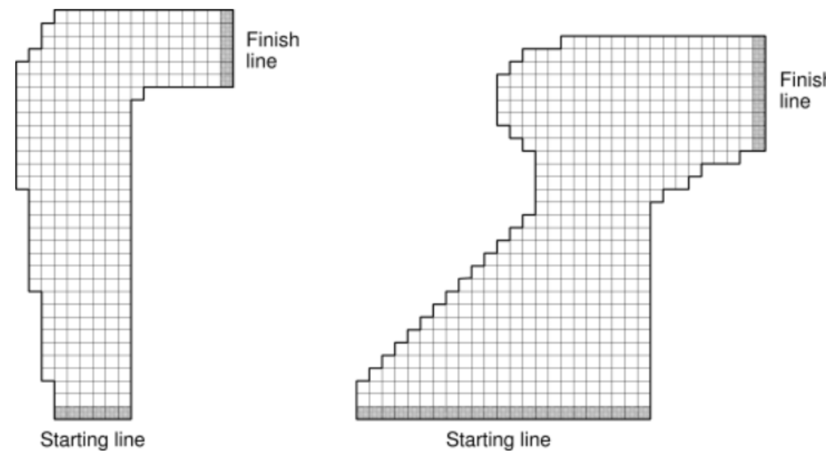
$$V_0 = W_0 = 0$$

incremental equivalent

We may also wish to assign relative weights to different episodes…

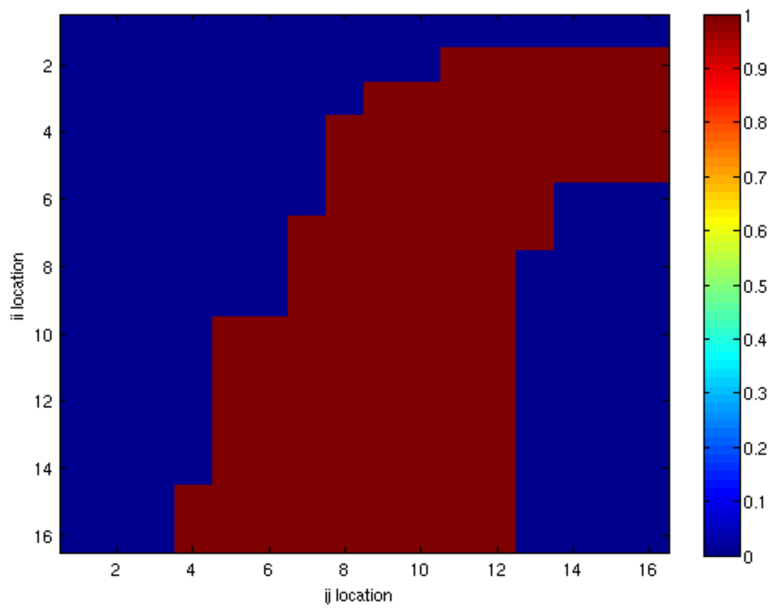
Racetrack Example

- Go as fast as possible but do not skid off the track
- Velocity = #grid cells (h/v) per time step, bounded
- Noise added to actions
- State/action space?
- Reward
- Episode?
- On-policy/off-policy learning?



Racetrack Example

Track Layout



State Value Function