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, $\varepsilon$-soft policies:

  – **On-policy** method: Explore in an $\varepsilon$-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 \( \varepsilon \)-soft:
  \[
  \pi(s,a) > 0, \forall s, \forall a
  \]
- Moreover, one may really wish to adopt an \( \varepsilon \)-greedy policy:
  \[
  \pi(s,a) = \frac{\varepsilon}{|A|}, \text{ if } a \text{ is not the greedy choice}
  \]
  \[
  = 1 - \varepsilon + \frac{\varepsilon}{|A|}, \text{ if } a \text{ is the greedy choice}
  \]
- In this case, we have
  \[
  \pi(s,a) > \frac{\varepsilon}{|A|}, \forall s, \forall a
  \]
On-Policy MC Control

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

Repeat forever:
(a) Generate an episode using $\pi$
(b) For each pair $s, a$ appearing in the episode:
- $R \leftarrow$ return following the first occurrence of $s, a$
- Append $R$ to $\text{Returns}(s, a)$
- $Q(s, a) \leftarrow$ average($\text{Returns}(s, a)$)
(c) For each $s$ in the episode:
- $a^* \leftarrow \text{arg max}_a Q(s, a)$
  
For all $a \in A(s)$:
- $\pi(s, a) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|A(s)| & \text{if } a = a^* \\ \varepsilon/|A(s)| & \text{if } a \neq a^* \end{cases}$

Evaluate as before

Improve towards $\varepsilon$-greedy, not the max
The Policy Improvement Step

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

\[
Q^\pi(s, \pi'(s, a)) = \sum_a \pi'(s, a)Q^\pi(s, a)
\]

\[
= \frac{\epsilon}{|A(s)|} \sum_a Q^\pi(s, a) + (1 - \epsilon) \max_a Q^\pi(s, a)
\]

\[
\geq \frac{\epsilon}{|A(s)|} \sum_a Q^\pi(s, a) + (1 - \epsilon) \sum_a \frac{\pi(s, a) - \epsilon}{1 - \epsilon} Q^\pi(s, a)
\]

This is bounded above by,

\[
= \frac{\epsilon}{|A(s)|} \sum_a Q^\pi(s, a) - \frac{\epsilon}{|A(s)|} \sum_a Q^\pi(s, a) + \sum_a \pi(s, a)Q^\pi(s, a)
\]

\[
= V^\pi(s)
\]
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: \( \pi(s, a) > 0 \iff \pi'(s, a) > 0, \forall s, \forall a \)
• 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 \( \pi \) and \( \pi' \)
Learning a Policy while Following Another

On the $i$th first visit to state $s$, let:

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

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

$R'_i(s) = \text{return observed from following the behaviour policy through this sequence of states and actions}$
Learning a Policy while Following Another

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

\[
p_i(s_t) = \prod_{k=t}^{T_i(s)-1} \pi(s_k, a_k) P_{s_k s_{k+1}}^{a_k}
\]

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

\[ p_i(s_t) = \prod_{k=t}^{T_i(s)-1} \pi(s_k, a_k) P_{s_k s_{k+1}}^{a_k} \]

\[ p_i'(s_t) = \prod_{k=t}^{T_i(s)-1} \pi'(s_k, a_k) P_{s_k s_{k+1}}^{a_k} \]

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

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* $\pi'$ to generate moves
  - must be soft so that all $(s, a)$ continue to be explored

- Evaluate and improve *Estimation Policy* $\pi$
  - converges to optimal policy

So...

1. BP $\pi'$ generates episode

2. EP $\pi$ is deterministic and gives the greedy actions w.r.t. the current estimate of $Q^\pi$ (it is arbitrary for the first episode)
Off-Policy MC Algorithm, cont.

3. Start at end of episode, work backwards

\[ \ldots \quad r_{T-1} \quad s_{T-1} \quad a_{T-1} \quad r_T \quad s_T \]

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

4. For this chain of states and actions compute

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

\( \pi \) is deterministic so \( \pi(s_k, a_k) \) etc. = 1 and we know \( \pi' \)
Off-Policy MC Algorithm, cont.

So

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

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 \(N\)

\(R' = \text{return for the chain of states/actions (see 3) following } (s, a) \text{ (it's different for each of the } N \text{ visits, as is } p/p')\)
6. Do for each \((s, a)\) in chain (see 3)

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

\[
\pi(s) = \text{arg max}_a Q(s, a)
\]

(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 \( s \in S, a \in A(s) \):
- \( Q(s, a) \leftarrow \text{arbitrary} \)
- \( N(s, a) \leftarrow 0 \) ; Numerator and
- \( D(s, a) \leftarrow 0 \) ; Denominator of \( Q(s, a) \)
- \( \pi \leftarrow \text{an arbitrary deterministic policy} \)

Repeat forever:
- (a) Select a policy \( \pi' \) 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) \( \tau \leftarrow \text{latest time at which } a_\tau \neq \pi(s_\tau) \)
- (c) For each pair \( s, a \) appearing in the episode after \( \tau \):
  \( t \leftarrow \text{the time of first occurrence (after } \tau \text{) of } s, a \)
  \( w \leftarrow \prod_{k=t+1}^{T-1} \frac{1}{\pi'(s_k, a_k)} \)
  \( N(s, a) \leftarrow N(s, a) + wR_t \)
  \( D(s, a) \leftarrow D(s, a) + w \)
  \( Q(s, a) \leftarrow \frac{N(s, a)}{D(s, a)} \)
- (d) For each \( s \in S \):
  \( \pi(s) \leftarrow \text{arg max}_a Q(s, a) \)
Incremental Implementation

• Better to implement MC incrementally (think memory...)

• To compute the weighted average of each return:

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

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

\[
W_{n+1} = W_n + w_{n+1}
\]

\[
V_0 = W_0 = 0
\]

non-incremental \hspace{1cm} \text{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