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 \( a \in A(s) \):
  - \( \pi(s, a) \leftarrow \begin{cases} 
    1 - \varepsilon + \varepsilon/|A(s)| & \text{if } a = a^* \\
    \varepsilon/|A(s)| & \text{if } a \neq a^*
  \end{cases} \)
Off-policy Monte Carlo control

- Behavior policy generates behavior in environment
- Estimation policy is policy being learned about
- Average returns from behavior policy by probability their probabilities in the estimation policy
Learning about $\pi$ while following $\pi'$

Suppose we have $n_s$ returns, $R_i(s)$, from state $s$, each with probability $p_i(s)$ of being generated by $\pi$ and probability $p'_i(s)$ of being generated by $\pi'$. Then we can estimate

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

(5.3)
Learning about $\pi$ while following $\pi'$

Suppose we have $n_i$ returns, $R_i(s)$, from state $s$, each with probability $p_i(s)$ of being generated by $\pi$ and probability $p'_i(s)$ of being generated by $\pi'$. Then we can estimate

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

(5.3)

which depends on the environmental probabilities $p_i(s)$ and $p'_i(s)$. However,

$$p_i(s_t) = \prod_{k=t}^{T_i(s)-1} \pi(s_k, a_k) P_{s_k s_{k+1}}^{a_k}$$
Learning about $\pi$ while following $\pi'$

Suppose we have $n_s$ returns, $R_i(s)$, from state $s$, each with probability $p_i(s)$ of being generated by $\pi$ and probability $p_i'(s)$ of being generated by $\pi'$. Then we can estimate

$$V^\pi(s) \approx \frac{\sum_{i=1}^{n_s} \frac{p_i(s)}{p_i'(s)} R_i(s)}{\sum_{i=1}^{n_s} \frac{p_i(s)}{p_i'(s)}}$$

which depends on the environmental probabilities $p_i(s)$ and $p_i'(s)$. However,

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

and

$$\frac{p_i(s_t)}{p_i'(s_t)} = \frac{\prod_{k=t}^{T_i(s)-1} \pi(s_k, a_k) \mathcal{P}_{s_k s_{k+1}}^{a_k}}{\prod_{k=t}^{T_i'(s)-1} \pi'(s_k, a_k) \mathcal{P}_{s_k s_{k+1}}^{a_k}} = \prod_{k=t}^{T_i(s)-1} \frac{\pi(s_k, a_k)}{\pi'(s_k, a_k)}.$$

Thus the weight needed, $p_i(s)/p_i'(s)$, depends only on the two policies and not at all on the environmental dynamics.
Reminder: On-policy control

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

\[
Q(s, a) \leftarrow \text{arbitrary} \\
Returns(s, a) \leftarrow \text{empty list} \\
\pi \leftarrow \text{an arbitrary } \epsilon\text{-soft policy}
\]

Repeat forever:

(a) Generate an episode using \( \pi \)

(b) For each pair \( s, a \) appearing in the episode:

\[
R \leftarrow \text{return following the first occurrence of } s, a \\
\text{Append } R \text{ to } Returns(s, a) \\
Q(s, a) \leftarrow \text{average}(Returns(s, a))
\]

(c) For each \( s \) in the episode:

\[
a^* \leftarrow \arg \max_a Q(s, a) \\
\text{For all } a \in \mathcal{A}(s):
\]

\[
\pi(s, a) \leftarrow \begin{cases} 
1 - \frac{\epsilon + \epsilon}{|\mathcal{A}(s)|} & \text{if } a = a^* \\
\frac{\epsilon}{|\mathcal{A}(s)|} & \text{if } a \neq a^*
\end{cases}
\]
Off-policy MC control

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

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 \( s \in S \):
\[\pi(s) \leftarrow \arg \max_a Q(s, a) \]
Summary

• MC has several advantages over DP:
  – Can learn directly from interaction with environment
  – No need for full models
  – No need to learn about ALL states
  – Less harm by Markovian violations (later in book)
• MC methods provide an alternate policy evaluation process
• One issue to watch for: maintaining sufficient exploration
  – exploring starts, soft policies
• Introduced distinction between on-policy and off-policy methods
• No bootstrapping (as opposed to DP)
• Mutalisk Strategy
• http://eis-blog.ucsc.edu/2010/10/starcraft-ai-competition-results
Chapter 6: Temporal Difference Learning

Objectives of this chapter:

- Introduce Temporal Difference (TD) learning
- Focus first on policy evaluation, or prediction, methods
- Then extend to control methods
TD Prediction

**Policy Evaluation (the prediction problem):**
for a given policy \( \pi \), compute the state-value function \( V^\pi \)

Recall:

Simple every-visit Monte Carlo method:
\[
V(s_t) \leftarrow V(s_t) + \alpha \left[ R_t - V(s_t) \right]
\]

\textbf{target:} the actual return after time \( t \)

The simplest TD method, TD(0):
\[
V(s_t) \leftarrow V(s_t) + \alpha \left[ r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \right]
\]

\textbf{target:} an estimate of the return
Simple Monte Carlo

\[ V(s_t) \leftarrow V(s_t) + \alpha [R_t - V(s_t)] \]

where \( R_t \) is the actual return following state \( s_t \).
Simplest TD Method

\[ V(s_t) \leftarrow V(s_t) + \alpha \left[ r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \right] \]
cf. Dynamic Programming

\[ V(s_t) \leftarrow E_\pi \{ r_{t+1} + \gamma V(s_t) \} \]
TD methods bootstrap and sample

- **Bootstrapping**: update involves an *estimate*
  - MC does not bootstrap
  - DP bootstraps
  - TD bootstraps

- **Sampling**: update does not involve an *expected value*
  - MC samples
  - DP does not sample
  - TD samples
# Example: Driving Home

<table>
<thead>
<tr>
<th>State</th>
<th>Elapsed Time (minutes)</th>
<th>Predicted Time to Go</th>
<th>Predicted Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>leaving office</td>
<td>0</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>reach car, raining</td>
<td>5 (5)</td>
<td>35</td>
<td>40</td>
</tr>
<tr>
<td>exit highway</td>
<td>20 (15)</td>
<td>15</td>
<td>35</td>
</tr>
<tr>
<td>behind truck</td>
<td>30 (10)</td>
<td>10</td>
<td>40</td>
</tr>
<tr>
<td>home street</td>
<td>40 (10)</td>
<td>3</td>
<td>43</td>
</tr>
<tr>
<td>arrive home</td>
<td>43 (3)</td>
<td>0</td>
<td>43</td>
</tr>
</tbody>
</table>
Driving Home

Changes recommended by Monte Carlo methods ($\alpha=1$)

Changes recommended by TD methods ($\alpha=1$)

[Graph showing predicted total travel time vs situation, with actual outcome marked at each step.]

R. S. Sutton and A. G. Barto: Reinforcement Learning: An Introduction
Advantages of TD Learning

• TD methods do not require a model of the environment, only experience

• TD, but not MC, methods can be fully incremental
  – You can learn before knowing the final outcome
    • Less memory
    • Less peak computation
  – You can learn without the final outcome
    • From incomplete sequences

• Both MC and TD converge (under certain assumptions to be detailed later), but which is faster?
Random Walk Example

Values learned by TD(0) after various numbers of episodes
TD and MC on the Random Walk

Data averaged over 100 sequences of episodes

R. S. Sutton and A. G. Barto: Reinforcement Learning: An Introduction