Introduction to Reinforcement Learning

Part 4: Monte Carlo Learning to solve Blackjack
Monte Carlo Methods for Solving MDPs

- Monte Carlo methods are *learning* methods
  - Experience $\rightarrow$ values, policy
- Monte Carlo methods can be used in two ways:
  - *On-line*: No model necessary and still attains optimality
  - *Simulated*: Planning without a full probability model
- Monte Carlo methods learn from *complete* sample returns
  - Only defined for episodic
Monte Carlo Policy Evaluation

- **Goal:** learn $v_\pi$
- **Given:** some number of episodes under $\pi$ which contain $s$
- **Idea:** Average returns observed after visits to $s$

- *Every-Visit MC:* average returns for *every* time $s$ is visited in an episode
- *First-visit MC:* average returns only for *first* time $s$ is visited in an episode
- Both converge asymptotically to $v_\pi$
First-visit Monte Carlo policy evaluation

Initialize:
\[
\pi \leftarrow \text{policy to be evaluated} \\
V \leftarrow \text{an arbitrary state-value function} \\
Returns(s) \leftarrow \text{an empty list, for all } s \in S
\]

Repeat forever:
(a) Generate an episode using \( \pi \)
(b) For each state \( s \) appearing in the episode:
\[
G \leftarrow \text{return following the first occurrence of } s \\
\text{Append } G \text{ to } Returns(s) \\
V(s) \leftarrow \text{average}(Returns(s))
\]
Backup diagram for Monte Carlo

- Entire episode included
- Only one choice at each state
- Time required to estimate one state does not depend on the total number of states
Blackjack example

- **Object:** Have your card sum be greater than the dealers without exceeding 21.
- **States** (200 of them):
  - current sum (12-21)
  - dealer’s showing card (ace-10)
  - do I have a useable ace?
- **Reward:** +1 for winning, 0 for a draw, -1 for losing
- **Actions:** stick (stop receiving cards), hit (receive another card)
- **Policy:** Stick if my sum is 20 or 21, else hit
Blackjack value functions

After 10,000 episodes

Usable ace

No usable ace

After 500,000 episodes

Dealer showing

Player sum
Monte Carlo Exploring Starts

Initialize, for all $s \in S$, $a \in A(s)$:
\[
\begin{align*}
Q(s, a) &\leftarrow \text{arbitrary} \\
\pi(s) &\leftarrow \text{arbitrary} \\
\text{Returns}(s, a) &\leftarrow \text{empty list}
\end{align*}
\]

Fixed point is optimal policy $\pi^*$

Now proven (almost)

Repeat forever:
(a) Generate an episode using exploring starts and $\pi$
(b) For each pair $s, a$ appearing in the episode:
\[
\begin{align*}
G &\leftarrow \text{return following the first occurrence of } s, a \\
\text{Append } G \text{ to } \text{Returns}(s, a) \\
Q(s, a) &\leftarrow \text{average} \left(\text{Returns}(s, a)\right)
\end{align*}
\]
(c) For each $s$ in the episode:
\[
\pi(s) \leftarrow \text{arg max}_a Q(s, a)
\]
Blackjack example continued

- Exploring starts
- Initial policy as described before
Monte Carlo Exploring Starts

Initialize, for all $s \in S$, $a \in A(s)$:
- $Q(s, a) \leftarrow$ arbitrary
- $\pi(s) \leftarrow$ arbitrary
- $Returns(s, a) \leftarrow$ empty list

Fixed point is optimal policy $\pi^*$

Now proven (almost)

Repeat forever:
(a) Generate an episode using exploring starts and $\pi$
(b) For each pair $s, a$ appearing in the episode:
   - $G \leftarrow$ return following the first occurrence of $s, a$
   - Append $G$ to $Returns(s, a)$
   - $Q(s, a) \leftarrow$ average($Returns(s, a)$)
(c) For each $s$ in the episode:
   - $\pi(s) \leftarrow \arg \max_a Q(s, a)$
Blackjack example continued

- Exploring starts
- Initial policy as described before

![Graphs showing optimal policies and values for Blackjack](image)
On-policy Monte Carlo Control

- **On-policy**: learn about policy currently executing
- How do we get rid of exploring starts?
  - Need soft policies: \( \pi(a|s) > 0 \) for all \( s \) and \( a \)
  - e.g. \( \epsilon \)-soft policy:
    \[
    \frac{\epsilon}{|A(s)|} \quad \text{non-max} \quad 1 - \epsilon + \frac{\epsilon}{|A(s)|} \quad \text{greedy}
    \]
- Similar to GPI: move policy *towards* greedy policy (i.e. \( \epsilon \)-soft)
- Converges to best \( \epsilon \)-soft policy
Summary for Monte Carlo Methods

- Monte Carlo methods learn directly from experience
  - *On-line*: No model necessary and still attains optimality
  - *Simulated*: No need for a full model
- Monte Carlo methods learn from complete sample returns
  - Only defined for episodic tasks
- The *Monte Carlo exploring starts* algorithm avoids the vexing issue of maintaining exploration