Monte Carlo Methods

Motivation

Last Lecture

- Solve a *known* MDP by dynamic programming
- **Solve** means answering the question *what shall I do in each state to maximize my expected discounted reward?*.
- We saw how to use knowledge of the environment for **prediction** of a policy, and **control** (optimize such policy).
  - **Prediction** means evaluation of the policy (how would the trajectories look like).
  - **Control** means finding the best policy.

Model-Based Prediction

- The **state-value function** $v_\pi(s)$ of an MDP is the expected return starting from state s, and then following policy π:

$$v_\pi(s) = \mathbb{E}(G_t \mid S_t = s)$$

- The **state-action value function** $q_\pi(s, a)$ is the expected return starting from state s, taking action a and then following policy π,

$$q_\pi(s, a) = \mathbb{E}_\pi(G_t \mid S_t = s, A_t = a).$$

Model-Based Control: Value Iteration.

Idea:

Use the optimality equation:

$$v_*(s) = \max_{a \in \mathcal{A}} \left\{ \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s') \right\}$$

to get a sequence:

$$v_{k+1} \leftarrow \max_{a \in \mathcal{A}} \left\{ \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_k(s') \right\}$$

Model-Based Control: Policy Iteration.

How to get close to optimal behaviour without calculating the value function

- Given policy π :
  - **Evaluate** π, that is, calculate $v_{\pi}(s)$ and/or Q_{π}.
  - **Improve** the policy by being **greedy** with respect to v_{π} and/or Q_{π}.
  - Update and repeat.
- Being greedy means, at state s, update the policy π by the policy π' such that:

$$\pi'(s) = \text{argmax}_{a \in \mathcal{A}} Q_{\pi}(s, a)$$

This process converges to the optimal policy π_*.

Model-Free Prediction

- Evaluate a policy without using explicit rewards or transitions from the underlying model.
- Part of the full reinforcement learning problem.

Model-Free Control

- **How to find the best policy?**
- Many problems can be modelled as MDPs, but either
  - MDP model is unknown, but experience can be sampled.
  - MDP model is known, but it is too big to use.
- In the model-free world, we use Q instead of v, since greedy policy improvement over Q does not depend on the model.

In this lecture:

- We will show how to adapt policy iteration in the model-free world.
- To substitute our knowledge of the world, we need to
  - Infer its reward and transition structure by repetition of many episodes.
  - Update our information "somehow" while running the episode.

Monte Carlo Learning

What are Monte Carlo methods?

- Key idea: Average sample returns at the end of each episode.

Assumptions

- Experience is divided into episodes.
- All episodes finish (eventually).

Monte Carlo Learning

- MC is model-free because we don't need to know the rewards or transitions of the underlying MDP.
- Estimates for one state do not "build upon" estimates of the other.
    - Useful if you only require the value at certain states.
- Learning from both real and simulated experience.
- **Might take too much time**, even in small problems.

Goal

- Learn Q_π from episodes of experience under π:

$$S_1, A_1, R_1, S_2, A_2, R_2, \ldots S_T \sim \pi$$

- Recall that the return G_t is

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{T-1} R_T$$

- The value function is the expected return

$$Q_\pi(s, a) = \mathbb{E}_\pi(G_t \mid S_t = s, A_t = a)$$

- In MC simulation we replace the expectation above by empirical mean.

Making it work

- To ensure that sampled average returns would converge to the value function, we need:
    - All episodes must start in a state-action pair.
    - All state-action pairs have positive probability of being selected at the start.
    - This guarantees that in the limit of an infinite number of episodes, all pairs would be selected infinitely many times.

Exploring starts

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

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

Repeat forever:
    Choose $S_0 \in \mathcal{S}$ and $A_0 \in \mathcal{A}(S_0)$ s.t. all pairs have probability > 0
    Generate an episode starting from S_0, A_0, following π
    For each pair s, a appearing in the episode:
        $G \leftarrow$ return following the first occurrence of s, a
        Append G to $Returns(s,a)$
        $Q(s,a) \leftarrow \text{average}(Returns(s,a))$
    For each s in the episode:
        $\pi(s) \leftarrow \arg\max_a Q(s,a)$

- MC with exploring starts has not been proven to converge!
- However, convergence is proven for a variation of this idea.

First-Visit Monte Carlo Policy Evaluation

- To evaluate a state s under a fixed policy π:
- Increment a counter the first that the pair s, a is visited in an episode

$$N(s, a) \leftarrow N(s, a) + 1.$$

- Increment total return $R(s, a) \leftarrow R(s, a) + G_t$
- Let $Q(s, a) \sim R(s, a)/N(s, a)$
- $Q(s, a) \rightarrow Q_\pi(s, a)$ as $N(s, a) \rightarrow +\infty$
- $\pi \rightarrow \epsilon - \mathrm{greedy}(\pi)$

Every-Visit Monte Carlo Policy Evaluation

- To evaluate a state s under a fixed policy π:
- Increment a counter every time that the pair s, a is visited in an episode

$$N(s, a) \leftarrow N(s, a) + 1.$$

- Increment total return $R(s, a) \leftarrow R(s, a) + G_t$
- Let $Q(s, a) \sim R(s, a)/N(s, a)$
- $Q(s, a) \rightarrow Q_\pi(s, a)$ as $N(s, a) \rightarrow +\infty$
- $\pi \rightarrow \epsilon - \text{greedy}(\pi)$

What's the difference?

- None in practice, theoretical analysis is different.
- Both methods are **on-policy**, meaning that they sample and evaluate from the same policy.

Incremental mean

The mean of a sequence $x_1, x_2, \ldots$ can be computed incrementally:

$$
\begin{aligned}
\mu_k &= \frac{1}{k} \sum_{j=1}^{k} x_j \\
&= \frac{1}{k} \left(x_k + \sum_{j=1}^{k-1} x_j \right) \\
&= \frac{1}{k} \left(x_k + (k-1)\mu_{k-1} \right) \\
&= \mu_{k-1} + \frac{1}{k} \left(x_k - \mu_{k-1} \right)
\end{aligned}
$$

Incremental Monte-Carlo Updates

- Update $Q(s, a)$ incrementally after each episode.
- For each state S_t with return G_t

$$
\begin{aligned}
N(S_t, A_t) &\leftarrow N(S_t, A_t) + 1 \\
Q(S_t, A_t) &\leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}(G_t - Q(S_t, A_t))
\end{aligned}
$$

- We can "forget the past" by compute an exponential moving mean. We don't move to correct the value all the way to the mean, we just correct it a bit.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(G_t - Q(S_t, A_t))$$

GLIE Monte-Carlo Control

- *Greedy in the Limite with Infinite Exploration* (GLIE)
- Sample an episode using policy π
- For each state S_t and action A_t in the episode,

$$
\begin{aligned}
N(S_t, A_t) \leftarrow \quad & N(S_t, A_t) + 1 \\
Q(S_t, A_t) \leftarrow \quad & Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}(G_t - Q(S_t, A_t))
\end{aligned}
$$

- Improve the policy based on the new state-action-value function
  - $\epsilon \leftarrow 1/k$
  - $\pi \leftarrow \epsilon - \text{greedy}(Q)$

Example: Frozen Lake

- https://gym.openai.com/evaluations/eval_TtcFIoaZQu6fGDIQICFKtw
- https://gist.github.com/jpmaldonado/fbc572b3bb517ac0848687b6e987f9a0

Off-policy

- Learn for a **different** policy to the one we are using to generate the episode.
- Off-policy methods consider both a
  - **target policy** t policy, from which we want to learn.
  - **behavior policy** b policy, from which we generate the episode.
- We need to guarantee **coverage** that is, $t(a|s) > 0$ implies $b(a|s > 0)$.

Importance sampling

- Estimation of expected values from a distribution given samples of another.

Weighted importance sampling

Off-policy MC control, for estimating $\pi \approx \pi_*$

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
  $Q(s,a) \leftarrow$ arbitrary
  $C(s,a) \leftarrow 0$
  $\pi(s) \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken consistently)

Repeat forever:
  $b \leftarrow$ any soft policy
  Generate an episode using b:
    $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T, S_T$
  $G \leftarrow 0$
  $W \leftarrow 1$
  For $t = T-1, T-2, \ldots$ downto 0:
    $G \leftarrow \gamma G + R_{t+1}$
    $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$
    $\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken consistently)
    If $A_t \neq \pi(S_t)$ then ExitForLoop
    $W \leftarrow W \frac{1}{b(A_t|S_t)}$

References

- David Silver's course (videos, slides).
- Sutton and Barto, *Reinforcement Learning: An Introduction*.